ACKNOWLEDGEMENTS

This thesis would not come to a successful completion without the help of the following:-

Kathy Yeung	Touran Ghahremani
Michael Mak	Edward Robert
José Pereda	D. Freeman
Riva Heft	Claude Pineault
A. Chaudhur	Lucy Pasillas
Alex Yongu	Michael Daigle
Helen Workman	Thomas Wilson
Eleitha Boykin	Laura Winer
L. Douglas	Claude LeBel
Anthony Dimicco	H. Thwaites
S. McDonnell	Armand Lauzon

I would like to express my appreciation to my friends Mark Sedgwick and Porter Scobey who have done a beautiful editorial job, and to Dr. Robert Jones who has given me a tremendous amount of help in running the main formative evaluation and his helpful comments. Special thanks should go to Dr. Gary Boyd for his supervision, excellent advice and solid comments, and Mr. Le Xuan, a very good friend and a teacher for his innovative ideas and critical viewpoints.

I also want to give special credit to a very special friend, Charles Soucy who helped from the beginning to the end of the thesis, served as a guinea pig of the experiment, did all the Xerox copies,

and provided me with spiritual support.

I am indebted to my friends, Jeannette Fong, Janson Fong, Wong Man Bing and my sister Theresa for their financial assistance during my study in Canada. Finally I want to acknowledge the patience and endurance of my mother who encouraged me in this work.

TABLE OF CONTENTS

TABLE OF FIGURES

SECTION 1: DEVELOPMENT AND EVALUATION OF AN INFORMATION MAPPED TEXT

(1) EDUCATIONAL PROBLEM

Most of the textbooks in the market are usually too wordy or too condensed. Many authors write textbook's haphazardly or according to their own wishes without giving the text a systematic instructional treatment. Unfortunately, even many instructional materials on instructional development found in the market do not practice what they preach (EPIE Report, 1978). Thus, it makes learning a difficult task to most of the learners.

The purpose of this thesis is to design, produce and evaluate a course unit about a computer language called 'BASIC' using an effective instructional technique called Information Mapping (Horn, 1969, 1971 & 1974). It aims at giving the subject matter BASIC a formal instructional treatment so that BASIC beginners can master the programming skills in BASIC after having studied the course unit.

(2) LITERATURE REVIEW

(A) Review of textbooks for the computer language BASIC

There were about 40 'BASIC' textbooks in the market after the first book about BASIC was published in 1966. Gray (1976) has reviewed 34 books about BASIC and their pitfalls are summarized as follows:-

(a) The biggest problem in those books is the inablity of the authors to put themselves in the readers' shoes, and write for the average beginners. Almost all authors ignore the fact that most of the readers are starting at zero and some assume the readers are as intelligent as the authors.

(b) Some authors cut their texts into very short chapters and others squeeze a number of concepts in one chapter.

(c) Some books have no standarization in arranging the content.

(d) Some authors begin the books by presenting 'BASIC' programmes but they have trouble to make a clear picture for the readers, or the programmes are too complicated or too long. Most of the programmes prove to be too complicated for many beginners.

(e) Most of the authors scatter the teaching of the elements of 'BASIC' throughout their books.

(f) The format of the books makes readability difficult.

(g) Some books do not have exercises. Some have overly long exercises. Some have exercises without answers.

Summing up, no formal instructional technique was used to design those books, therefore, it makes computer programming difficult to learn for most of the learners. In view of the above, there is a real need to write a 'BASIC' book by using a strong and effective

instructional technique. Hence, the development of this Information Mapped text is expected to solve the above problems.

(B) Literature review on Information Mapping

(a) The nature of Information Mapping

Information Mapping is an instructional technique invented by Horn (1969, 1971 & 1974). The technique is a synthesis of the "current learning research, instructional technology" and good communication practice "into a comprehensive materials development and presentation technology to improve technical communication"[1], learning and teaching. The new tool is hailed as a method to overcome the paper mountain (Horn, 1974). The method outlines a systematic way to present printed information according to its type. It has been considered as a breakthrough in print format (Thiagarajan, 1977) that provides a tool in instructional design to make communication easier and quicker. The information presented has no paragraphs and unnecessary traditional phrases. The method is unique in its way to organize, write, sequence and format learning and reference materials. Its main merit is to cut through textbook verbiage, therefore, Information Mapping is also considered as a viable alternative to lengthy and complex prose (Hlynka, 1979).

(b) The Components of Information Mapping

(i) The format of Information Mapping

The design of a book has a strong impact on readers (Williamson, 1966). Spencer (1968) found out a number of factors can determine the legibility of a text e.g. the grid, the format, the typography, etc. Macdonald-Ross & Waller (1975) showed that a standardized grid can achieve consistency, clarity and order in printed information. A well-designed typography can solve the layout problem and it provides visual consistency and pleasure in reading (Dair, 1967). In case of presenting a large complex body of information, a standardized format is indispensable and the material must be presented in a well-organized fashion to facilitate learning (Flavell, 1963).

A number of the above features have been implemented in the format of Information Mapping. The information in Information Mapping is broken down into chunks to ensure a meaningful organization and presentation (Stafford & Combs, 1967) so that they can provide easy retention and recall. The information is arranged in such a standardized typography and format that the headings and sub-headings are easily skimmed and the readers are able to find out quickly where they are. A systematically arranged Information Mapped text gives a straightforward path to readers without any risk of getting lost or confused in a large and complex material. Thus, the readers can have

an easy access to all chunks of information in the instructional material and the information can communicate in the easiest possible way. In a nutshell, the overall purpose of Information Mapping is to contribute pleasure in reading, increase reading efficiency and enhance learning.

(ii) Types of Maps and Blocks

A unit in Information Mapping can be a procedure, structure, concept, process, classification or summary of facts. These six basic types of maps are constructed from some 40 Information Blocks. Only Blocks and Maps are used in Information Mapping instead of chapters and paragraphs. The Block becomes the counterpart of the paragraph, and Map the chapter. However, the Blocks, unlike paragraphs, are labelled to identify the kind of information they present. Therefore, the information presented by Information Mapping is specific and to the point.

All the materials in Information Mapping are arranged in a hierarchy (Horn, 1971):-

1st: Courses

2nd: Units or sections

3rd: Information Maps

4th: Information Blocks

The smallest chunk of an Information Map is an Information Block. An Information Block consists of:-

- Sentence(s) (and/or diagram(s)) about a logical fragment of a subject matter, and
- a label which functionally describes the content of an Information Block e.g. "Definition", "What is it?", "Example", "Non-example", "Note", "Related Concepts", etc.

Information Blocks are easily identifiable because they are separated by horizontal lines and their labels are conspicuously displayed in the margin. All Information Blocks are similar in structure but not in content. An Information Block defines the information it presents and there is no overlapping of information that does not belong to that particular Block. Because of its format and nature, all Information Blocks are modular since they can be replaced, taken out, improved, shortened and lengthened.

Basic Information Maps (which contain information new to the learners):-

An Information Map (a lesson or a chapter in its conventional counterpart) consists of a cluster of Information Blocks about a particular topic. An Information Map is always displayed at the top of a page. There are six basic types of Maps in Information Mapping.

(1) Concept Maps (which define and give examples and non-examples of a new concept),

(2) Structure Maps (about the physical things and objects

which have identifiable boundaries),

(3) Process Maps (which explain how processes or operations work; how changes take place in time),

(4) Procedure Maps (which explain how to do things and in what order to do them),

(5) Classification Maps (which show how a collection of concepts is organized), and

(6) Fact Maps (which give results of observations or measurements).

Supplementary Maps

Horn (1976) also uses a list of supplementary maps to aid learning:

- Initial Learning Information Maps
  e.g. Overview Maps, Review Maps, Summary Maps, Prerequisites Map, Compare & Contrast Map, Course Objectives, Learning Advice Map, etc.
- Reference Information Maps
  e.g. Table of Contents, Index, etc.
- Exercise & Questions Maps
  e.g. Self Text, Posttest, Pretest, etc.

There are no mandatory blocks used in supplementary maps.

(c) The differences between Information Mapped text and other texts (1972, Whitlock)

- There is no paragraph in Information Mapped text as compared with traditional text. There is no frame as compared with programmed instruction. Paragraphs and frames are replaced by Information Blocks (chunks of information) and Information Maps (collection of Blocks). A book written in Information Mapping consists only Maps and Blocks.
- There are specific rules to write the types of Blocks and Maps.
- There is a classification of Blocks and Maps in basic Maps (i.e. seven Maps and some 40 basic types of Blocks).
- Information Mapping is totally modular. Additional information can be added without disturbing the overall integrity of the text.
- The information is presented in a logical and orderly way.
- There is a visible format in Information Mapped text and its layout is its outstanding feature.

(d) Theories behind Information Mapping

The principle of Information Mapping is drawn from accumulated theories of different fields like instructional design, education, communication, learning, psychology and graphic design. It uses a number of things such as task analysis, behavioral objectives, criterion-referenced test, prerequisites test, etc. The followings are some of the researches which are implemented into the principle of Information Mapping:-

- Feedback often facilitates learning by confirming or correcting learners' understanding (Lumsdaine & May, 1965; Gagné & Rohwer, 1969).

  The above leads to the formations of "Exercise Map" & "Answer Map" in Information Mapping.

- Self test and pretest facilitate retention (Glaser, 1965; Briggs, 1968; Bloom, 1963).

  The above leads to the formations of "Self Test Map" & "Pretest Map" in Information Mapping.

- In concept learning, a variety of examples and non-examples promotes learning (Gagné & Rohwer, 1969; Lumsdaine, 1963; Merrill & Tennyson, 1977).

  The above leads to the formations of "Example Block" and "Non-example Block" in Information Mapping.

- Instructions are helpful in calling learner's attention to important features (Gagné, 1970; Gagné & Rohwer, 1969). The above leads to the formations of "Introduction Map" and "Preview Map" which serve as advance organizers (Ausubel, 1968).
- "Cueing" or labelling seems to help by alerting learner to the nature of the upcoming information and inform him what the learning task is (Glaser, 1965). This leads to the use of marginal labels and informative map titles in Information Mapping.

(e) Applicability of Information Mapping

Information Mapping is a method to categorize and display information (Horn, 1971). Its applicability includes the development of textbook and technical, training and reference manuals by business and industry. It also has been applied in documenting company policies and procedures. Olympia (1979) found that it can be used to write textbook on chemistry and other related sciences. It is particularly useful for writing computer languages (Whitlock, 1972; Horn, 1976) and self-instructional textbooks. Many large coperations and university departments in U.S. and Great Britain have implemented the technique and reported significant gains in instructional efficiency (Horn, 1976).

(f) Limitations of Information Mapping

Though Information Mapping has a plethora of applications it is not a panacea for all. Because of its rigid format, the text tends to be choppy and there are a number of areas that its applicability is difficult (Horn, 1974; 1976):-

- simulation, gaming and role-playing,
- proposals or project planning,
- history or reporting on what has been happening in the organization,
- not to be used to teach psychomotor skills,
- not to be used to improve motivation, and
- thesis.

(g) Studies done in Information Mapping

Literature shows only two experimental studies were done in the area of Information Mapping. The first study was carried out by Geisert (1970) who compared an Information Mapped text (based on the early version of Information Mapping in 1969) with a traditional text on 44 members of the Army National Guard of Tallahassee. The subjects were found to show a more positive attitude toward the Information Mapped material than the traditional text and their performance suggested that Information Mapped treatment resulted in better achievement. Another study done by Jonassen (1979) was to compare an Information

Mapped text (based on the latest version in 1976) with a programmed text on the learner's ability to facilitate recall and retrieval of information. The results showed both methods produced very significant scores, however, no differential effect was found. According to these studies, the method of Information Mapping appears to be effective and to have a great potential in the design of instructional material.

(C) Literature review on the design and evaluation of instructional material

There are many models for the development of instructional material (Stolovitch, 1976; Thiagarajan, 1976). One common model is called 'system approach'(Dick & Carey, 1978; Kemp, 1971; Popham & Baker, 1970). The model consists of a number of components that outline the procedures for the design of instructional material (Briggs, 1970; Briggs, 1977; Bruner, 1966; Friesen, 1973; Gagné & Briggs, 1974). The emphasis of this approach is to plan and organize the instructional material through a number of interconnected steps so that the designer can design, produce, evaluate and revise the instructional material in order that the material can achieve its desired outcome:-

(a) Stating an educational problem

This step focuses on the identification of the problem area

and the justification of the existence of the problem (Friesen, 1973; Kaufman, 1972). The problem has to be clearly identified and it can be solved through a task analysis.

(b) Choosing a medium

This is the decision made to select systematically the appropriate medium for the instructional material (Anderson, 1976; Gerlach & Ely, 1971; Romiszowski, 1974; Rowntree, 1974). Its cost and availability are examined and its possible functions are sought to ensure that it can maximize learning. The selection of the medium should be based on the behavioral analysis of the instructional problem and its potential for implementing specific objectives (Gerlach & Ely, 1971) rather than merely the merits of the medium (Romiszowski, 1974).

(c) Identifying an instructional goal

This step determines what the learners will be able to do after the completion of the course. The goal is the synthesizing or culminating behavior of the terminal skill (Davies, 1976; Mager, 1972) which is based on the higher levels of knowledge and understanding. The instructional goal has to be directly related to the content of the course designed (Gagné & Briggs, 1974).

(d) Conducting an 'instructional' analysis

This step determines the types of skills to be taught to the learners as related to the general goal. The anslysis usually

outlines all the necessary subordinate and superordinate skills in a hierarchical fashion (Cook & Walbesser, 1973; Gagné & Briggs, 1974). The acquisition of the subordinate skills promotes the transfer of the learning of the superordinate skills. Therefore, in designing instruction, the instructional sequence should be arranged according to an hierarchy of knowledge to promote learning (Gagné, 1970).

(e) Identifying the target population and prerequisites

This step involves describing for whom the material is designed. The target population is described in terms of age, sex, education, etc. The entry behaviors of the target population have to be specified before they can start the instructional activity (Davis, Alexander & Yelon, 1975; DeCecco, 1968).

(f) Writing behavioral objectives

The behavioral objectives are derived directly from the instructional analysis (Dillman & Rahmlow, 1972). They describe the clear, precise and overt skills or behaviors that the learners will be able to do after the completion of the course (Davies, 1976; Mager, 1975). Though the nature of behavioral objectives has been questioned as inedequate and unrealistic in measuring some capabilities e.g. latent ones (Mitchell, 1972), it is still widely advocated by a number of people (Briggs, 1977; Gagné & Briggs, 1974; Kibler, Cegala, Barker & Miles, 1974). One common model of behavioral objective includes the following components:-

- observable skills or behaviors which the learner will be able to do,

- the conditions, and
- the criteria or mastery level.

(g) Developing criterion-referenced tests

This step is the design of tests to measure the behavioral objectives of the instructional material (Payne, 1974; Popham, 1973). The constructed tests are used to determine the outcome of the learners' performance. The commonly used criterion-referenced tests are entry behaviors test, pretest, posttest, embedded test, etc (Dick & Carey, 1978; Gagné & Briggs, 1974).

(h) Designing and conducting formative evaluation

Formative evaluation is the process to gather data from the criterion-referenced tests to improve the effectiveness of the instructional material (Baker & Alkin, 1973; Baker & Saloutos, 1974; Bloom, Hastings & Madaus, 1971; Popham, 1973). Usually, there are three stages in formative evaluation: one-to-one, small group and field evaluation (Dick & Carey, 1978).

(i) Analyzing the data of formative evaluation

This step lists the method to interpret the data of formative evaluation. There are many ways to analyze the data e.g. error matrix (Horn, 1976), histogram, interquartile range (Parson, 1974 ; Runyon-Haber, 1971), etc. The main purpose of this process is to find the difficulty of the instructional material so that revision can be

conducted accordingly (Morris & Fitz-Gibbon, 1978; Popham, 1975).

(j) Conducting summative evaluation

The purpose of summative evaluation is to assess the effectiveness of the instructional material so that a decision can be made to adopt the material to be used by other learners or teachers (Dick & Carey, 1978; Popham, 1975). The information can be based on the formative evaluation to back up its position (Fitz-Gibbon & Morris, 1978). In order to substantiate the success or the failure of the course, believable information and acceptable evidence are collected to attest the merits of an instructional material.

(k) Conclusion

The above model summarizes all the necessary procedures to design instructional material. It is made up of interactive components which bring about feedback to the system. Its intent is to improve the overall developmental process of the instructional material.

(3) REASONS FOR CHOOSING INFORMATION MAPPING AS A MEDIUM

Information Mapping is chosen as an instructional technique in producing the text under the following grounds:-

- It is an effective instructional design technique.
- It is an effective method for the design of self-

instructional material

- It improves the print format by providing easy scanning, readability, initial learning, reference and review.
- It helps to organize and sequence the format of learning and instructional materials.
- It promotes learning by its clear order and organization of the materials.
- It cuts down lengthy and complex prose.
- It is applicable for writing computer languages.
- The technique is supported by learning and psychology theories.

(References: Hlynka, 1979; Horn, 1969, 1971, 1974, 1976; Olympia, 1979; Thiagarajan, 1977; Whitlock, 1972)

(4) MAPS AND BLOCKS USED IN WRITING THIS COURSE UNIT

Two kinds of map are used in writing this instructional course unit: (A) Concept Map, and (B) Supplementary Maps.

(A) Concept Map

The concept map is used to introduce a new term (or topic)

that is not part of the learners' regular vocabulary or whose usage is peculiar to its situation (Horn, 1976). Psychologists often say that a person has learned a new concept if he/she can define it or he/she can generalize within a class of things and can distinguish the members of the class from other classes. Hence, a concept map usually consists of the following blocks:-

(a) Name of Map,

(b) Introduction,

An Introduction Block is used as an advance organizer for the information to be presented in the upcoming map. It serves to sensitize readers for what he/she will be learning and it contributes to the continuity of the Information Mapping materials.

(c) Definition (What is it?),

A Definition Block is used when a new term is introduced and we feel our readers do not understand. The Definition Block defines or describes the concept being introduced.

(d) Example,

An Example Block gives an instance of the concept being presented.

(e) Non-example,

A Non-example Block gives a negative example of the

concept being presented.

(f) Rule (Theorem or Generalization),

A Rule Block is used when we want to provide a general statement for inference, conception or principle that gives general applicability to a particular concept.

(g) Note (Comment),

A Note Block is used to present any additional information that might be helpful but which cannot be sorted out into any of the Information Block categories.

(h) Exercises,

An Exercises Block is used to give exercises activities to learner the concept taught.

(i) Answers, etc.

An Answer Block is used to give response (feedback) to the exercises given.

In this course unit the following Maps are written in Concept Map (refer to Section 2):-

Map Number	Name of Map
1	What is BASIC?
4	Line Number
5	The Name (Keyword) of a Statement
7	Statement Syntax: Constant

Map Number	Name of Map
8	Statement Syntax: Numerical Constant
9	Statement Syntax: String Constant
10	Statement Syntax: Internal Constant
11	Statement Syntax: Variable
12	Statement Syntax: Numerical Variable
13	Statement Syntax: String Variable
15	Statement Syntax: Operator
16	Statement Syntax: Relational Operator
17	Statement Syntax: Punctuation Marks
18	REM STATEMENT
19	PRINT STATEMENT
20	PRINT TAB STATEMENT
21	LET STATEMENT
22	INPUT STATEMENT
23	READ & DATA STATEMENTS
24	END STATEMENT
25	SYSTEM COMMAND: RUN
26	SYSTEM COMMAND: LIST

The following supplementary maps are used in the course unit:-

(a) Course Objective Map

A Course Objective Map is used to list a number of behavioral objectives which the students can accomplish after the completion of the course unit.

(b) Overview (Preview) Map

An Overview Map is used to aid learning and it organizes the material of a unit or chapter in advance. It is used for the beginning of each section. Two Overview Maps are used in this course unit:-

Map Number	Name of Map
2	Overview of BASIC
6	Overview of the Statement Syntax

(c) Compare & Contrast Map

A Compare & Contrast Map is used when the difficulty of learning occurs due to the confusion of two similar concepts. Its main purpose is to promote discrimination learning between the two confusing concepts. Two Compare & Contrast Maps are used in this course unit:-

Map Number	Name of Map
3	Compare & Contrast Table on: System Commands & BASIC Statements
14	Compare & Contrast Table on: Numerical & String Variables

(d) Self Test Map

A Self Test Map is a test which is administered by the learner. It enables the learner to determine how well he knows a particular chunk of information in a course. Branching is provided so that the learner can skip the part of material that he knows. Almost every map in this course unit is preceded with a Self Test Map.

(e) Pretest Map

A Pretest Map is a test used at the beginning of this course unit. It consists of questions which are the major information taught in the course unit. If the learner passes the pretest at a stated level he/she can skip the whole course unit. If the learner fails the pretest he is advised to study the unit.

(f) Posttest Map

A Posttest Map is a test which is given after the learner has completed the course unit. It consists of the questions which are the major skills and knowledge taught in this course unit. It is used to determine whether the learner

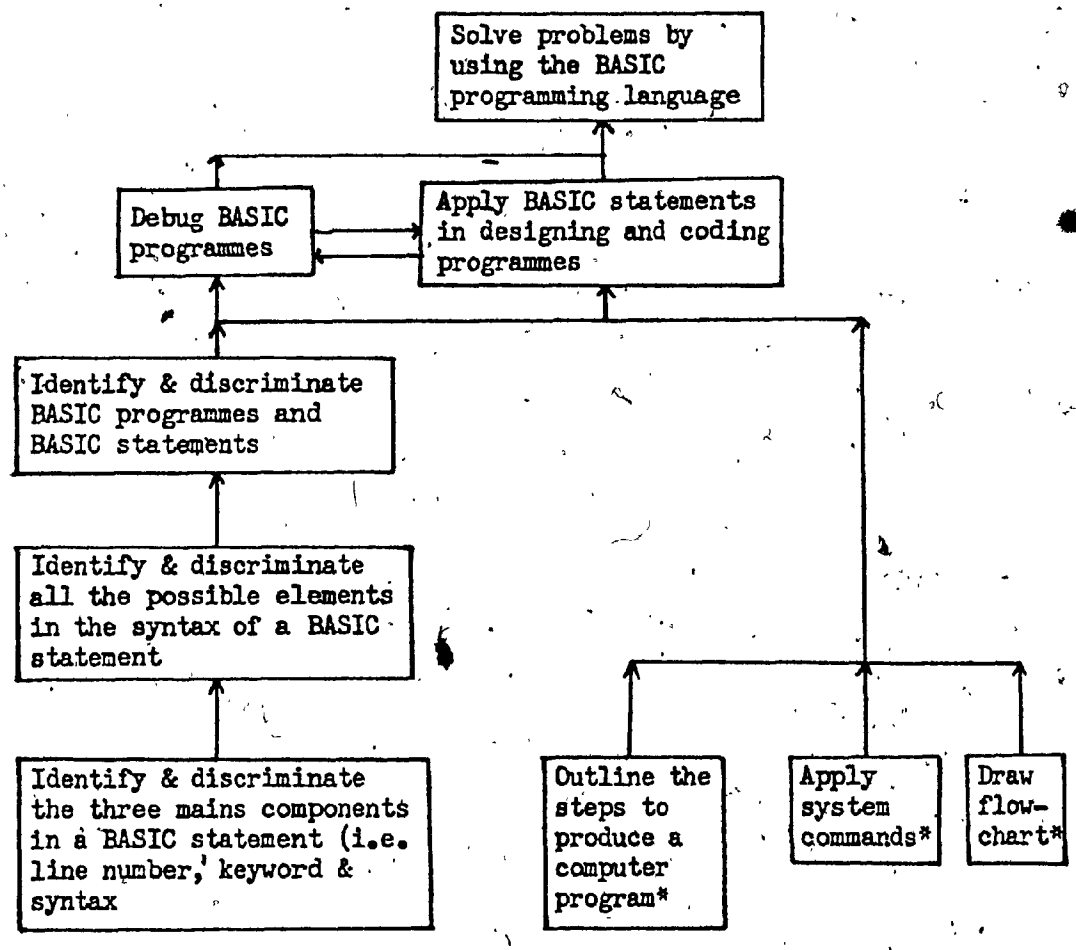
has learned the material in the course unit.

(5) INSTRUCTIONAL GOAL

The general goal of this course unit is to enable the learners to have the knowledge, skills and understanding in designing and coding programmes in 'BASIC' programming language. The content treatment of the course unit, pretest and posttest are directly related to the instructional goal.

(6) INSTRUCTIONAL ANALYSIS

The instructional approach in the next page outlines the procedure of the relevant skills which the learners are expected to achieve the instructional goal. This is an analysis that breaks the parts of the task and relates the parts to each other (Silvern, 1970). This describes how to do a task in an organizational procedure. The procedure identifies and specifies the subject topics to be taught. It can be seen that the skill of applying the 'BASIC' programming language in problem solving is the superordinate skill. In order to acquire this skill other subordinate skills have to be mastered as prerequisites so as to facilitate positive transfer.



Instructional (task) Analysis of BASIC usage

*Note: These skills are to be taught by the lecturer, therefore, they are not included in this Information Mapped text.

This instructional analysis identifies all the necessary skills, concepts and information that need to be learned by the learner and it also provides a logical sequence for writing the instruction.

(7) TARGET POPULATION AND PR REQUISITES

This course unit is designed for 'BASIC' beginners. The target population is the general public with high school education. They are expected to be able to type since they have to go to the computer terminal to use the keyboard. They are also supposed to have some motivation and interest in learning how to master 'BASIC' programming technique. In the main-evaluation operation the subjects were Educational Technology or Diploma students who were taking the course called 'Small Computer Systems for Teachers and Trainers'.

(8) BEHAVIORAL OBJECTIVES

The behavioral objectives are derived directly from the instructional analysis. They describe the skills and behaviors that the learners are able to do after the completion of the unit:-

After having studied this Information Mapped text, you will be able to score at least 90% in the posttest.

(a) At the completion of this course, you will be able to design and code simple programmes by using BASIC programming language.

(b) You will be able to apply the following BASIC statements in designing and coding programmes:- REM, PRINT, PRINT TAB, LET, INPUT, READ, DATA & END

(c) You will be able to list the rules and logic in using the above-mentioned BASIC statements.

(d) You will be able to detect and debug bugs in some simple BASIC programmes.

(e) You will be able to identify and discriminate the basic features of a BASIC program.

(f) You will be able to identify and discriminate the components and the sub-components of a BASIC statement.

It should be noted that the behavioral objectives are designed as a learning hierarchy (Bloom, 1964; Davies, 1976). That is to say the lowest level starts from understanding of knowledge (terms, facts, rules, etc.), then it advances to the higher level of application, analysis and problem solving.

(9) CRITERION-REFERENCED TESTS

(A) Pretest (Appendix 1)

The pretest is designed to test whether the learners have the knowledge in 'BASIC' or not. If the learner scores more than 90% (the mastery level) in the pretest it shows that he/she knows the subject matter well and he/she is not required to learn the course unit. If he/she scores less than 90% he/she is advised to study the course unit.

The test items designed in the pretest are parallel to the behavioral objectives stated on page 26. Since the main objectives are the skill of designing and coding programmes in 'BASIC', therefore, this type of question in the pretest is given the highest value in scoring. But the lower level of skill like identifying and discriminating terms, facts and rules (e.g. the basic features of a 'BASIC' program, the components and the sub-components of a 'BASIC' statement) is given low value in scoring. The marking scheme for the pretest is enclosed at Appendix 3.

(B) Posttest (Appendix 2)

The posttest is equivalent to the nature of the pretest.

They both are constructed to measure the stated behavioral objectives. The content validity of the posttest items is guaranteed by its close correspondence to the behavioral objectives set out on page 26 (no reliability statistics were used due to the heterogeneous nature of the criterion items). The posttest also emphasizes the super-ordinate skill of the learners (i.e. to solve problem by designing and coding 'BASIC' programmes), therefore, problem solving type test item is allocated with 66% of the total score (the highest percentage of the total mark). The marking scheme for the posttest is at Appendix 4.

(10) FORMATIVE EVALUATION

Formative evaluation was conducted once the instructional material had been prepared in its first draft. This kind of evaluation is commonly used to try out the course unit with the representative learners in order to gather information and data which would be used to revise, improve and redesign the instructional material. The emphasis is to make the instructional unit as effective as possible. Two phases of formative evaluation were carried out in this course unit.

(A) First phase: One-to-one evaluation

(a) Participation by learners from the target population

This evaluation was conducted on the basis of one learner

from the target population and the designer on the draft of the course unit. The pretest, the instructional course unit, the posttest and the course evaluation form were given to the learner separately. The learner was encouraged to point out the deficiencies, the strengths and the problems of the course unit and the instructional technique. The designer observed the results carefully, noting the points where the learner experienced difficulty, confusion and irritation. The learner was interviewed to determine where the problems were, and how the instructional material and the instructional technique should be improved and revised. The followings were types of informations the designer was looking for in this one-to-one evaluation:-

- faulty instructional analysis,
- faulty wording or unclear passages,
- inadquate information presentation:-
  - wrong presentation of maps and blocks,
  - unclear examples and non-examples,
  - too much or too little information given,
  - examples and non-examples too abstract,
  - wrong sequence of presentation,
  - wrong exercises and answers, etc.

- unclear test questions or test directions,
- unclear or inappropriate behavioral objectives,
- grammatical mistakes, etc.

The following is a summary table showing the performance of the four learners in one-to-one evaluation:-

Table 1

Learners' Performance in One-to-one Evaluation

	Learner	Pretest	Posttest	Total hours spent on the course unit
1	government officer	0	65	5 hours
2	undergraduate student	0	81	8 hours
3	graduate student	*	*	did not finish
4	mathematics professor	*	*	6 hours

*Note: Test was not taken.

All the learners had no knowledge in 'BASIC' but learner number 2,3 & 4 had some programming experience. They were the representative of the target population. The course unit was given to them to study separately and the time spent was marked. They were all interviewed and made lots of informative comments. Their reaction helped to find out typing, grammatical and some other errors. The mistakes were corrected. The graphic cues (i.e. the arrows and underlyings used for illustrative purposes in the course unit) were standardized and redone in order to enhance learning through emphasis and attention focusing. The first map called "What is BASIC?" was rewritten and lengthened to give a more global viewpoint. Change of content, deletion of ambiguous parts and change of presentation order were also done. Because of all these changes, the whole course unit had to be retyped.

(b) Participation by instructional expert

This part of one-to-one evaluation was conducted between a conceptual analyst and the designer. The analyst was

given the rough draft of the first version and had provided the designer a lot of feedback in the following areas:-

- instructional analysis,
- the treatment of the content,
- the presentation of information chunk,
- the clarification of the instructional technique (i.e. Information Mapping), etc.

(c) Participation by subject matter expert

The lecturer who gave the course called 'Small Computer Systems for Teachers & Trainers (ETEL563)' in Educational Technology program is an acknowledged subject matter specialist in the area of the course unit and had verified the accuracy of the content. The expert also offered valuable comments on the general instructional strategy in the unit especially the idea of "Self Test".

(B) Second phase: Main-evaluation operation

After the course unit had been revised on the basis of the information obtained from the one-to-one evaluation, the main formative evaluation was conducted. The subjects

were 22 students enrolled in Educational Technology or Diploma Program in Concordia University. They were taking the course called 'Small Computer Systems for Teachers and Trainers (ETEL 563)' from June 16 to July 17, 1980. The students were supposed to be motivated to learn the material in the course unit because the score of the posttest was worth 15% of the total mark of the course ETEL 563. The pretest was administered on the first day of the class and the time spent was one hour. The Information Mapped text was handed out to each subject for self-study for the period from 16th June to 26th June. They were told the nature of the study and encouraged to comment on the strengths and the limitations of the unit. The posttest was given on 26th June and the time spent was one hour and 15 minutes. The purpose of pretest and posttest was to determine whether there would be an improved performance on the subjects after the study of the course unit. The course unit evaluation form (evaluation questionnaires)(Appendix 5) was given to find out students' reactions towards the instructional course unit and the instructional technique used. This helped to determine whether the course unit was effective with

the target population and to identify changes in the unit to improve its effectiveness.

(11) COLLECTION AND DISCUSSION OF EVALUATION DATA

Data which would help to make decisions about improving the instruction were collected:-

- Test data collected on pretests and posttests,
- Comments made by the students marked on the course unit about its strengths and difficulties,
- Course unit evaluation questionnaires and debriefing comments given by the students on the overall reactions to both the course unit and the instructional technique (i.e. Information Mapping), and
- Feedback given by the subject matter specialist and conceptual analyst.

The main purpose of data collection in formative evaluation is to determine the effectiveness of the course unit. The following tools are used for analysing the data:-

(A) Tools for analysing data

(a) Score distribution (Table 2)

A score distribution is a presentation of the distribution of test scores. It assesses the overall performance of a group of

learners and it is useful for overall assessment. Table 2 (page 36) shows the score distribution of the overall student performance in the pretest and the posttest. All the students did better in the posttest than the pretest. Since the mastery level is 90%, 8 learners reached the mastery level and the rest did not.

(b) Interquartile range (Table 2)

Interquartile range is used to measure the dispersion of score data. The range is simply calculated by subtracting the first quartile value (i. e., 25th percentile) and the third quartile value (i.e., the 75th percentile). In the posttest, the 25th value is 95 and the 75th value is 43.5, therefore, the interquartile range is 41.5%. This shows there is a moderate spread in the achievement in the course unit. It also indicates that the course unit worked for some students but not others.

(c) Error matrix

An error matrix is a student-by-item display that shows the performance of each learner on each item of a test. It is used

Table 2

Score Distribution:

Student Performance in Pretest & Posttest

	Learner Number	Pretest 100%	Posttest 100%
	1	79	100
	2	3	99
	3	20.5	98
	4	24.5	96
75th percentile	5	0	95
	6	34.5	92.5
	7	25	92.5
	8	10.5	91
	9	36.5	88
	10	13.5	88
	11	2	79
	12	22.5	66
	13	12	63
	14	10	60.5
25th percentile	15*	0	43.5
	16	4.5	34
	17	0	32.5
	18	0	27
	19	12	26
	20*	0	4.5

Note: 90% is the mastery level.

* drop-out

to analyse the performance of the individual learner in relation to the behavioral objectives. The purpose is to evaluate the learning problems of each learner.

Table 3 (page 38) and table 8 (page 39) show the error matrices of both the pretest and the posttest. The error matrix of pretest displays all learners are the perfect subjects of the target population because no one reached the mastery level of 90%. The error matrix of posttest exhibits that the kind of objectives or skills the learners achieved after they had studied the course unit.

(d) Histogram (Graph 1)

A histogram is a graph chart showing the relative frequency of the entire discussed data set. It is very useful for statistical inference. Graph 1 is a histogram showing the total hour spent by each student on the Information Mapped text, the total hour spent by each student on the reference books, the pretest and the posttest scores. It summarizes graphically the overall performance of all the learners in respect to the time they spent on both the Information Mapped text and the reference books.

Table 3
Error Matrix of Pretest

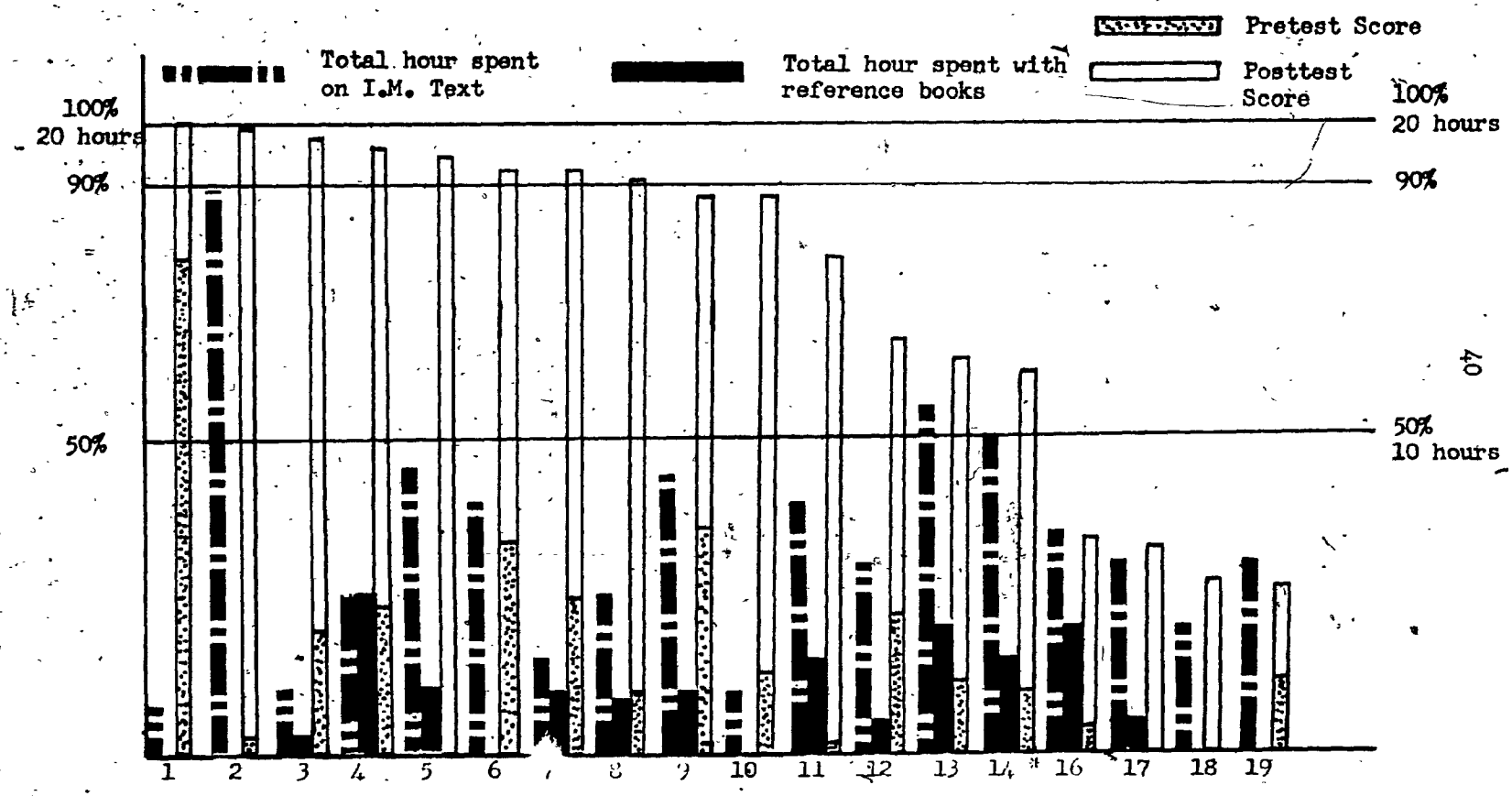
	Test Item																																														Total Score
	Part 1																				Part 2										Part 3		Part 4	Part 5										
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	1	2		1	2	3	4	5	6	7	8	
1					-5														-5												-2	4						+0			-6	75
2				-5	-5			-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5		-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	3
3		-5																-5				-5							-5	-5	-8	-3	-2			-4	-10	-10	-12	-12	-12	24.5
4					-5																		-5				-5		-5	-5	-6	-3	-2	-4		-2	-10	-10	-12	-12	-12	24.5
5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0
6		-5									-5												-5						-5	-5	-2	-3	-2	-4		-4	-10	-10	-12	-12	-4	34.5
7		-5	-5		-5	-5	-5				-5	-5	-5			-5	-5				-5	-5	-5						-5	-5	-6	-6	-1	-4	-2	-4	-10	-10	-12	-12	-12	13.5
8											-5	-5						-5				-5	-5			-5	-5		-5	-5	-3	-3	-5	-4		-4	-10	-10	-12	-12	-12	25
9		-5	-5	-5	-5	-5	-5	-5	-5		-5		-5	-5		-5	-5	-5	-5	-5	-5	-5	-5	-5					-5	-5	-2.5	-6	-4	-4	-2	-4	-10	-10	-12	-12	-12	10.5
10		-5	-5			-5								-5							-5		-5						-5	-5	-4.5	-6	-3	-4		-2	-7	-10	-7	-4	-12	36.5
11		-5		-5	-5			-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	2
12		-5			-5											-5					-5		-5								-8	-2	-1	-4		-4	-10	-10	-12	-12	-12	27.5
13		-5				-5		-5	-5	-5		-5	-5	-5							-5	-5	-5	-5					-5	-5	-8	-6	-3	-4		-4	-10	-10	-12	-12	-12	12
14		-5			-5	-5	-5				-5	-5	-5				-5				-5	-5	-5	-5					-5	-5	-8	-6	-3	-4	-2	-4	-10	-10	-12	-12	-12	10
15	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0
16					-5	-5	-5		-5					-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0.5
17	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0
18	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0
19		-5			-5	-5		-5			-5		-5		-5		-5	-5	-5		-5	-5	-5	-5	-5	-5			-5	-5	-8	-6	-1	-4		-4	-10	-10	-12	-12	-12	12
20	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-8	-6	-5	-4	-2	-4	-10	-10	-12	-12	-12	0
Score of each item	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	8	6	5	4	2	4	10	10	12	12	12	

Table 4
Error Matrix of Posttest

Test Item	Part 1																				Part 2										Part 3			↓	Part 4 Part 5							Total Score
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5	6	7	8	9	10	1	2	3		1	2	3	4	5	6	7	
1																																										100
2																																		-1								99
3																																									-2	98
4							-1																											-2								96
5											-.5							-1												-.5	-3											95
6				-1							-1				.5																	-3		-1	-1								92.5
7				-1		-1	-1											-1												-.5		-1		-1					-1	-1			92.5
8				-1																												-1		-.5	-2								91
9				-1																											-4			-2							-5		88
10				-1																											-4		-5	-2									88
11				-1				-1											-1												-4		-1	-6	-1			-2			-4		79
12			-1								-1			-1					-1	-1							-.5		-.5		-5	-1	-2	-5	-5		-.5	-2	-4	-4	-4		66
13				-1		-1																												-8				-2	-12	-12	-1		63
14				-1																											-5	-1	-5	-4				-2	-4	-12			60.5
15	-1			-1				-1			-1			-1	-1	.5	-1	-1									-.5		-.5		-9	-1	-1	-6			-2	-1	-8	-8	-1		53.5
16				-1				-1			-1			-1	-1	-1	-1	-1							-.5	-.5	-.5	-.5	-.5	-.5	-7	-5	-2	-8	-2		-3	-8	-10	-10			34
17				-1	-1		-1				-1			-1													-.5	-.5		-.5	-4	-1	-6	-7	-2		-4	-8	-12	-12	-5		32.5
18				-1				-1			-1																				-.5	-.5	-6	-8	-2		-4	-8	-9	-12	-12		27
19				-1			-1							-1	-1	-1	-1	-1								-.5			-.5	-5	-5	-.5	-4	-7	-2		-2	-7	-10	-12	-12		26

Graph 1

Frequency Distribution of Learners' Score in Posttest & Pretest, Hours Spent on Text & Reference Books



*Note: Learner 15: drop-out

(B) Discussion of data

The pretest shows learners: 2,5,8,10,11, 15,16,17 & 20 had no knowledge in 'BASIC' while learners: 3,4,6,7,9,12,13,14 & 19 had very little knowledge in 'BASIC'. Eight learners (i. e. learners 1 to 8) achieved the mastery level of 90% in the posttest. They mastered the skills intended to be taught in the course unit as shown in the error matrix of the posttest:-

- They mastered the skill in designing and coding simple programmes by using BASIC programming language.
- They are able to apply the following BASIC statements in coding programmes:-

  REM, PRINT, PRINT TAB, LET, INPUT, READ, DATA & END
- They know how to apply rules and the logic in using the above-mentioned statements.
- They are able to identify and discriminate the basic features of a BASIC program.
- They are able to identify and discriminate the components and the sub-components of a BASIC programmes.

They have occasional trouble in debugging BASIC programmes.

They completed studying the entire course unit. The maximum time spent was 18 hours (learner number 2) and the mininum time spent was 2 hours (learner number 3). Three of them spent no time on reference books (learner 1,2 & 6). Three learners spent 2 hours on reference book(s), one 5 hours and one half an hour.

The rest of the learners scored less than 90%. Learners 9 & 10 (scored 88) are the borderline cases. Only learners 9, 10,11,12 & 13 finished studying the entire course unit but the rest of the learners did not finish the whole unit (learner 12 stopped at page 50, 16 at page 80, 17 at page 88, 18 at page 98, and 19 at page 90). Two learners: 15 & 20 dropped out of the course. Two learners spent no time on reference book(s) (learners 18 & 19) and the rest did (4 hours, 1 hour, half an hour, 2 hours and half an hour). Learners 13 & 14 come from Latin America and their mother tongue is not English. Learner 16's mother tongue is not English, either.

Since learners 9 and 10 could almost reach the level of mastery (90%) they almost mastered all the skills in the posttest except debugging. The rest of the learners mastered some of simple concepts in 'BASIC' programming i.e. identifying and discriminating the basic features of a 'BASIC' program, the

components and the sub-components of a 'BASIC' statement. Their trouble area was in designing and coding simple programmes by using the 'BASIC programming language especially in applying the following statements:-

PRINT, INPUT, READ & DATA

It also shows that the related skill was affected e.g. debugging 'BASIC' programmes. Even though the posttest matrix shows these learners improved as a result of the course unit their gains are not adequate for mastery.

(12) INTERPRETATION OF THE FORMATIVE EVALUATION DATA

Factors affecting the learning outcomes

- The Information received during the class's period

During the class period the lecturer covered only the following materials: the history of computer, the nature of computer and flow-charting which are not the content material of the course unit. Therefore, it is concluded the information given by the lecturer has no direct effect on the learning outcomes of the posttest.

- The time spent on the reference book(s)

The learners were asked what kinds of information they looked up when they were reading textbook and reference book(s). Some disclosed that most of the information was related to the class assignments and the material they referred to was not the content

material of the course unit. It was checked that the textbook used contains a very small section in 'BASIC' programming. However, the amount of time they spent on the textbook (or reference books) is not much and its impact is considered minimal because the total time spent by all learners is 20.5 hours with an average of 1.2 hours per learner (but 108 hours was spent on the course unit by all learners with an average 6.3 hours per learner). Therefore, this factor is not considered to be a determining one that would give rise to their improvement in learning.

The main reason why 8 learners achieved the behavioral objectives is the fact that they all finished studying the course unit. The couse unit is a self-contained instructional material which is designed to promote learning. The perfect examples are the learners: 2 & 5 who had no previous knowledge in 'BASIC' and did tremendously well after having studied the course unit. It shows that the course unit has a positive effect on learning. This also gives supporting evidence showing the instructional material is effective.

The main reason why the rest of the learners could not reach the mastery level is due to their failure to finish the entire course unit and the reasons for their failure to finish the unit were numerous (e.g. heavy course load, personal committment with other business, not enough time given, etc.). If they did not study the

whole course unit it is logical that they could not answer all the questions in the posttest and failed to achieve the behavioral objectives.

Another determining factor is the language problem for non-English speaking learners. Two learners (from Latin America) had finished studying the course unit but still failed to reach the mastery level because they have some language problems. If they spent a little more time they might have performed better.

Interpretation of course unit evaluation

Table 5 summarizes the students' responses on the attitudes towards the course unit and the instructional technique Information Mapping. Their reaction is extracted from the course unit evaluation form (Appendix 5). The measurement used for the question numbers 6 to 15 and 21 is a 5-point Likert scale where 5 usually denotes "excellent" and 1 "very poor". The table shows the distribution of their responses and the mean scores. Standard deviation cannot be used to measure dispersal of the scores because the sample size is not big enough to entertain a normal distribution (Note: It needs more than 30 samples to have a normal distribution.)

Table 5

Learners' Reaction Towards

The Text & The Instructional Technique

	Total Learner: N=13	
	Criterion Group N=8	Others N=5
2. How much time did you spent on the course unit?	See Graph 1	See Graph 1
3. Did you use other books as reference?	Yes: 5 No: 3	Yes: 3 No: 2
4. If yes, how many books did you use? Please write down the name(s) of the book(s).	o The Applesoft Tutorial, o CAL Using BASIC o BASIC by Data General o Personal Computing: A Beginner's Guide	
5. If yes, how much time did you spend on the book(s)?	See Graph 1	See Graph 1

	Distribution of Answers (-)				(+)	Mean Score
	1	2	3	4	5	
6. Are the behavioral objectives clearly stated in the course unit?				8	5	4.3

7. Is the content of the course closely related to the behavioral objectives?				7	6	4.5
8. Is the content organized to promote learning?	1		1	6	5	4.0
9. Does the text provide a clear picture of the subject treated?	1			5	7	4.3
10. Is the language used clear and understandable?		2	2	3	6	4.0
11. Are the examples and non-examples in the course unit helpful in explaining the concepts concerned?			2	5	6	4.3
12. Does the format of the course unit (e.g. the label and the horizontal lines) encourage easy scanning?				6	7	4.5
13. Are the graphic cues (e.g. arrows and underlinings) helpful in enhancing learning through emphasis and attention focusing?			1	6	6	4.3
14. Do you prefer the format of this course unit to the traditional textbook?			5	3	5	4.0
15. Is the course unit helpful as self-instructional material?	1			4	8	4.3
21. On the whole, what do you think about the course unit?	1		1	8	3	3.9

Evaluation items 16, 17, 18, 19, 20 & 22 are open questions and they are summarized as follows:-

Strengths of the course unit stated by the learners

- The course unit provides basic features about 'BASIC' programming.
- It is easy to learn as a self-instructional material.
- It is interesting and challenging to learn.
- It is a clear and well structured instruction for beginners.
- The sequential arrangement of the material is most helpful.
- It provides a good progression from simple to complex material.
- It is well broken down into small steps.
- It gives confidence to the learners.
- The material is easier to be assimilated than the material of the traditional textbook.
- It is a great assistance to the Educational Technology course.
- It is well researched.

- The examples are clear and straightforward.
- The best part is the explanation of "variable", "string" and "constant".

Limitations of the course unit stated by the learners

- There are occasional flaws in grammar and typing.
- It is somewhat repetitive.
- There is no branching.
- It is too simplistic.
- There are not enough exercises.
- It is limited to beginners.

(d) Revision after the formative evaluation

Typing and grammatical mistakes were corrected. A self-test map was put in front of almost every map and branching technique was provided. More exercises were added to give more student participation. Some areas in content were changed.

It cannot be denied that the course is somewhat repetitive. However, it should be noted that the course is designed for beginners and its repetitiveness is based on the assumption that the beginners would have more exposure of the important aspect of the material while their memory would be reinforced. Sometimes, repetition is a good instructional strategy. Since branching is implemented in the

course unit, the fast learners can skip and avoid the repetitiveness.

The Information Mapped text has gone through several revision from the first draft to its last version. After the advice of the conceptual analyst, the first draft was revised twice. The thired version was revised after two learners had tried out in one-to-one formative evaluation. The fourth version was further revised after the main formative evaluation operation. The fifth version was carried out after the comment of the subject matter expert. Its latest version includes the editoral work of two English teachers.

SUMMARY OF OVERALL GAIN OF THE FINAL PRODUCT

The following information summarizes the general picture of the final product of the Information Mapped text:-

- The time required to develop this unit was about 11 man months.
- The estimated cost of photocopying the course unit is $18.00 but printed in quantity the cost would be reduced appreciably.
- No special teacher training is required.
- Nature of the text:-
  - The material is up-to-date.
  - There are tests and exercises to measure skill acquisition.

- The material is validated by the subject matter expert.
- The learner is informed what is going to be achieved in the objectives.
- The learner is free to go as quickly or as slowly as he/she wishes because the unit is self-paced. Branching is provided.
- Merits of using this material as indicated by the evaluation (stated by the learners):-
  - It develops student's interest because of its simple format.
  - It has a logical sequence of instruction.
  - The material is easily readable.
  - The material is thoroughly researched.
  - It is good for beginners.
  - The unit is entirely self-instructional.
  - It gives confidence to learners.

CONCLUSION AND RECOMMENDATIONS

The Course Unit

The Information Mapped text has been repeatedly revised to improve its effectiveness. This experiment showed that the text

had a positive impact on the learning of those who completed it. Nevertheless, it is plausiable that the sub-criterion group might have performed much better if they had worked through the whole text. The time factor appears to be the main one which determines the success of students' performance. Based on the learning outcome of the learners, their observed attitudes and other parts of the formative evaluation, it is concluded that the text has reached a considerable level of success in terms of its effectiveness. It would be interesting to arrange for a broader trial to confirm the apparent effectiveness shown with this small sample.

Cost Effectiveness

The cost effectiveness of a course is the analysis of its cost and achievement (Haller, 1974; Harmon, 1970; Hartley, 1968). It provides a decision tool to select the best choice among the feasible alternatives on the basis of the least cost and the greatest effectiveness. Hence, in order to determine the cost effectiveness of this course unit, two factors have to be examined:-

- its cost as compared with others, and
- whether the course unit has achieved its behavioral objectives or not, as compared with other courses.

The estimated cost in producing this course unit is about $24,000.00 (which covers the expenses for the designer, photocopying, research and developing costs, etc.) while the market price would need to be about $8.00 which is the average price of other textbooks in the market. It would be fully economic if twenty or thirty thousand of the copies were sold to recover the cost to make a reasonable profit. As mentioned in the review of 'BASIC' textbooks, most of the books do not reach learners' expectation and they have troubles in attaining their objectives (as a matter of fact most textbooks do not have any behavioral objectives). This Information Mapped text, as concluded before, has achieved its effectiveness to a great degree. Therefore, in order to determine the pay-off function of this course unit, it can also be concluded that it has a beneficial effect in terms of its cost effectiveness.

Why Information Mapping is Effective?

Horn's concept of Information Mapping is unique in its nature and structure. Its 6-map categories encompass a wide range of information types. It also provides a valuable scheme for classifying information. In Information Mapping the nature of information is analyzed, classified and organized by its type (such as: concept, procedure, process, classification, fact and structure). The idea of different kinds of 'Block' is used to break down a 'Map' or a

topic into very small chunks of information. Hence, Information Mapping provides a regular and consistent pattern so that information is assembled (or synthesized) to serve a meaningful purpose. This framework makes the small chunks of information fit together so that ideas are easily memorized and learned. It appears Information Mapping exemplifies some psychological underpinnings in the human thinking process because it helps to organize and structure our thought, and it tends to complete the thinking of an idea by linking the small chunks of information into a meaningful whole. Its underlying power as an analysis and synthesis tool seems to contribute to its effectiveness as an instructional technique through its far-reaching implications and generality of application.

Recommendations

Horn has shown that Information Mapping is an effective instructional technique and the method has been implemented to develop a variety of instructional materials. Up to now only two experimental studies have been carried out in this area and they both showed Information Mapping had good effectiveness. In this present study the technique is also rated favourably because of its capacity for increasing learning and retention. Nevertheless,

further research is recommended to compare Information Mapped texts with programmed texts (or even a traditional text) to justify its effectiveness. Other types of questions which can be asked in the future studies are: When is Information Mapping most appropriately used? Are there any underlying factors that determine the use of Information Mapping? Can Information Mapping extend its applicability to other than its present 6-map categories and 40 block types? Can Information Mapping be otherwise further improved?

BIBLIOGRAPHY

Anderson, R.H. Selecting and developing media for instruction. New York: Van Nostrand Reinhold Co., 1976.

Ausubel, D.P. Educational psychology: a cognitive view. New York: Holt, Rinehart & Winston, 1968.

Baker, E.L. & Alkin, M.C. Formative evaluation of instructional development. Audio Visual Communication Review, 1973, 21, no. 4.

Baker, E.L. & Saloutos, A. Evaluating instructional program. Los Angeles: UCLA, Graduate School of Education, 1974.

Bloom, B.S. Testing cognitive ability and achievement in Gage N.L. (ed.). Handbook of research on teaching. Chicago: Rand-McNally, 1963.

Bloom, B.S. Taxonomy of Educational objectives: the classification of educational goals. New York: David Mckay Co., Inc., 1964.

Bloom, B.S., Hastings, J.J. & Madaus, G.F. Handbook on formative and summative evaluation of student learning. New York: McGraw Hill Book Co., 1971.

Briggs, L.J. Learner variables and educational media. Review of Educational Research, April, 1968, 38, no. 2.

Briggs, L.J. Sequencing of instruction in relation to hierarchies of competence. American Institutes for Research, 1968.

Briggs, L.J. Handbook of procedures for the design of instruction.

Pittsburge, Pa.: American Institutes for Research, 1970.

Briggs, L.J. Instructional design. Englewood Cliffs, N.J.: Educational Technology Pub., 1977.

Bruner, I.S. Toward a theory of instruction. Cambridge: Harward University Press, 1966.

Cook, J.M. & Walbesser, H.H. How to meet accountability with behavioral and learning hierarchies. College Park, Md.: Bureau of Educational Research & Field Services, College of Education, University of Maryland, 1973.

Dair, C. Design with type. Toronto: University of Toronto Press, 1967.

Davies, I.K. Objectives in curriculum design. London: McGraw-Hill Book Co. (UK) Ltd., 1976.

Davis, R.H., Alexander, L.T. & Yelon, S.L. Learning systems design. N.Y.: McGraw Hill, 1975.

DeCecco, J.P. The psychology of learning and instruction: educational psychology. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1968.

Dick, W. & Carey, L. The systematic design of instruction. Glenview, Illinois: Scott, Foresman & Co., 1978.

Dillman, C.M. & Rahmlow, H.F. Writing instruction objectives. Belmont, California: Lear Siegler, Inc./Fearon Pub., 1972.

Educational Products Information Exchange Institute. Analyses of

materials for developers of instructional materials (EPIE Report). N.Y.: EPIE Instutute, vol. XI, no. 4M, summer, 1978.

Fitz-Gibbon, C.T. & Morris, L.L. How to design a program evaluation. Beverly Hill, California: Sage Pub., The Rgents of the University of California, 1978.

Flavell, J.H. The developmental psychology of Jean Piaget. Princeton, N.J.: Van Nostrand, 1963.

Friesen, P.A. Designing Instruction. Santa Monica, California: Miller Pub. Co., Educulture, Inc., 1973.

Gagné, R.M. The conditions of learning (2nd ed.). New York: Holt, Rinehart & Winston, 1970.

Gagné, R.M. & Briggs, L.J. Principles of instructional design. New York: Holt, Rinehart & Winston Inc., 1974.

Gagné, R.M. & Rohwer, W.R. Instructional psychology. Annual Review of Psychology, 1969, 20, pp. 381-418.

Geisert, P. A comparison of the effects of Information Mapped learning materials and traditional materials on the learning of concepts via the printed page and computer cathode ray tube. Washington, D.C.: Office of Naval Research, Personnel & Training Research Programs Office, October, 1970.

Gerlach, V.S. & Ely, D.P. Teaching media: a systematic approach. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1971.

Glaser, R. Training research and education. N.Y.: Wiley, 1965.

Gray, S.B. Creative computing feature review: 34 books on BASIC in D.H. Ahl. The Best of Creative Computing, March 1976, 1 & 2, no. 2.

Haller, E.J. Cost analysis for program evaluation: in evaluation in education - current applications. Berkeley, Cal.: Macuthan Pub., 1974.

Harmon, P. Curriculum cost-effectiveness evaluation, Audiovisual Instruction, Janunary 1970, 15, no. 1, pp. 24-26 & 76-77.

Hartley, H.J. Educational planning-programming-budgeting. Englewoood Cliffs, N.Y.: Prentice-Hall, 1968.

Hlynka, B. Four single-page learning models. Audiovisual Instruction, April 1979, 24, pp.20-21.

Horn, E.R. Information mapping for learning and reference. Cambridge, Mass.: Information Resources, Inc., 1969.

Horn, E.R. A reference collection of rules and guidelines for Information Mapped materials. Cambridge, Mass.: Information Resources, Inc., 1971.

Horn, E.R. Information Mapping for computer-based learning reference. Cambridge, Mass.: Information Resources, Inc., 1971.

Horn, E.R. The Information Mapping method of analytical writing. Training in Bussiness & Industry, March 1974, 11, no. 3.

Horn, E.R. Information Mapping: new tool to overcome the paper mountain. Educational Technology, May 1974, 14, pp. 5-8.

Horn, E.R. How to write Information Mapping. Lexington, Mass.: Information Resources Inc., 1976.

Jonassen, D.H. Recall and retrieval from mapped and programmed text. Paper presented at the Annual Convention of the Association for Educational & Technology, New Orleans, Louisiana, March 1979.

Kaufman, R.A. Educational system planning. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1972.

Kelly, J. Organizational behavior. Homewood, Illinois: Richard D. Irwin, Inc., 1969.

Kemp, J.E. Instruction design: a plan for unit and course development. Belmont: Fearon Pub., 1971.

Kibler, R.J., Cegala, D.L., Baker, L.L. & Miles, D.T. Objectives for instruction and evaluation. Boston: Allyn & Bacon, 1974.

Likert, R. A technique for the measurement of attitudes. Archives of Psychology, 1932, 140, pp. 1-55.

Lumsdaine, A.A. Instruments and media of instruction in Gage, N.L. (ed.). Handbook of research on teaching. Chicago: Rand-McNally, 1963.

Lumsdaine, A.A. & May, M.A. Mass communication and educational media. Annual Review of Psychology, 1965, 16, pp. 475-534.

Macdonald-Ross, M. & Waller, R. Criticism, alternatives, and test. London: The Institute of Educational Technology, The Open University, Sweet & Maxwell, Ltd., 1975.

Mager, R.F. Goal analysis. Belmont, California: Fearon Pub., Inc., 1972.

Mager, R.F. Preparing instructional objectives (2nd ed.). Belmont, California: Fearon Pub., Inc., 1975.

Merrill, M.D. & Tennyson, R.D. Teaching concepts: an instructional design guide. Englewood Cliffs, N.J.: Educational Technology Pub., Inc., 1977.

Mitchell, P.D. The sacramental nature of behavioral objectives. In Aspects of educational technology. Vol. VI, Pitman Pub., Association for Programmed Learning & Educational Technology, 1972.

Morris, L.L. Fitz-Gibbon, C.T. Evaluator's handbook. Beverly Hills: Sage Pub., The Regents of the University of California, 1978.

Olymia, P.L. Jr. Information-Mapped chemistry. Journal of Chemical Education, March 1979, 56, pp. 176-78.

Parsons, R. Statistical analysis: a decision-making approach. N.Y.: Harper & Row, Pub., 1974.

Payne, D.A. The assessment of learning: cognitive and affective. Lexington, Mass.: D.C. Heath & Co., 1974.

Popham, W.J. Evaluating instruction. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1973.

Popham, W.J. Educational evaluation. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1975.

Popham, W.J. & Baker, E.L. Systematic instruction. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1970.

Romiszowski, A.J. The selection and use of instructional media: a system approach. London: Kogan Page Ltd., 1974.

Rowntree, D. Educational technology in curriculum development. London: Harper & Row Pub., 1974.

Runyon, R.P. & Haber, A. Fundamental of behavioral statistics (2nd ed.). Reading, Mass.: Addison-Wesley Pub. Co., 1971.

Silvern, L.C. Systems Engineering of Education VI: principles of computer-assisted instruction system. Los Angeles, Cal.: Education & Training Consultants Co., 1970.

Spencer, H. The visible word. N.Y.: Visual Communication Books Hastings House, Pub., 1968.

Stafford, K.R. & Combs, C.F. Radical reductionism: a possible source of inadequacy in autoinstructional techniques. American Psychologist, 1967, 22, pp. 667-669.

Stolovitch, H. Are systems approach models useful? A definite maybe. NSPI Journal, 1976, 15(10), pp. 5-7.

Thiagarajan, S. Help, I am trapped inside an ID model! Alternatives to the systems approach. NSPI Journal, 1976, 15(9).

Thiagarajan, S. Information Mapping: a threakthrough in print format. Audiovisual Instruction, September 1977, 22, pp. 16-17.

Whitlock, Q.A. An interview with Robert E. Horn, the originator of Information Mapping. Industrial Training International, August 1972,

7, pp. 232-234.

Williamson, H. Methods of book design (2nd ed.). London: Oxford University, 1966.

FOOTNOTE

Note 1: Horn, R.E. How to write Information Mapping. Lexington, Mass.: Information Resources Inc., 1976, p.2.

Section 2: THE INFORMATION MAPPED TEXT

UNDERSTANDING AND APPLYING BASIC

Concordia University

Montreal, P.Q. Canada

1980

67

TABLE OF CONTENTS

PART III: BASIC STATEMENTS

REFERENCE

PREFACE

Introduction

This is a course unit taken from the book called Understanding & Applying BASIC. The instructional technique used is called Information Mapping (R.E. Horn, 1976). Each lesson (or chapter) is called a Map. There is no paragraph but a Block which consists of a sentence (or sentences) between 2 horizontal lines.

Audience

This course unit is designed for people who want to master BASIC programming language.

Prerequisites

(1) high school education
(2) Be able to type

Learning Advice

(1) There is a Self Test before each Map. If you are a beginner, ignore the self tests. If you have some programming experience, try the self test which determines how well you know the material of the map. It enables you to skip those materials that you know.

(2) Cover the answer section when you are working on the questions and check the answers afterwards.

(3) Active participation is necessary for learning BASIC programming. Go to the terminal and work out all the exercises. It will clear a lot of your doubts. Remember: practice makes perfect.

PART I: THE ELEMENTS OF BASIC

(*Note: The page number of the Information Mapped text is bracketed at the bottom of the page while the thesis page number is centred on the top of the page.)

SELF TEST

Note

This test enables you to determine how well you know the material in the map on page 4. If you can answer the following question correctly, go to page 18. If not, go to page 4. When you are working on the question cover the answer section. Check your result afterwards.

Exercise

Identify the 3 basic parts (i.e. line number, the keyword of a statement, and the syntax of a statement) in the following BASIC statements:-

```
10 REM This is a program to compute the
20 REM squares of some numbers.
30 LET A=1
40 PRINT "The square of ";A; " equals "; A*A;" ."
```

Answer

```
10 REM This is a program to compute the
```

This is the syntax of this REM statement.

The word 'REM' is the keyword (name) of this statement.

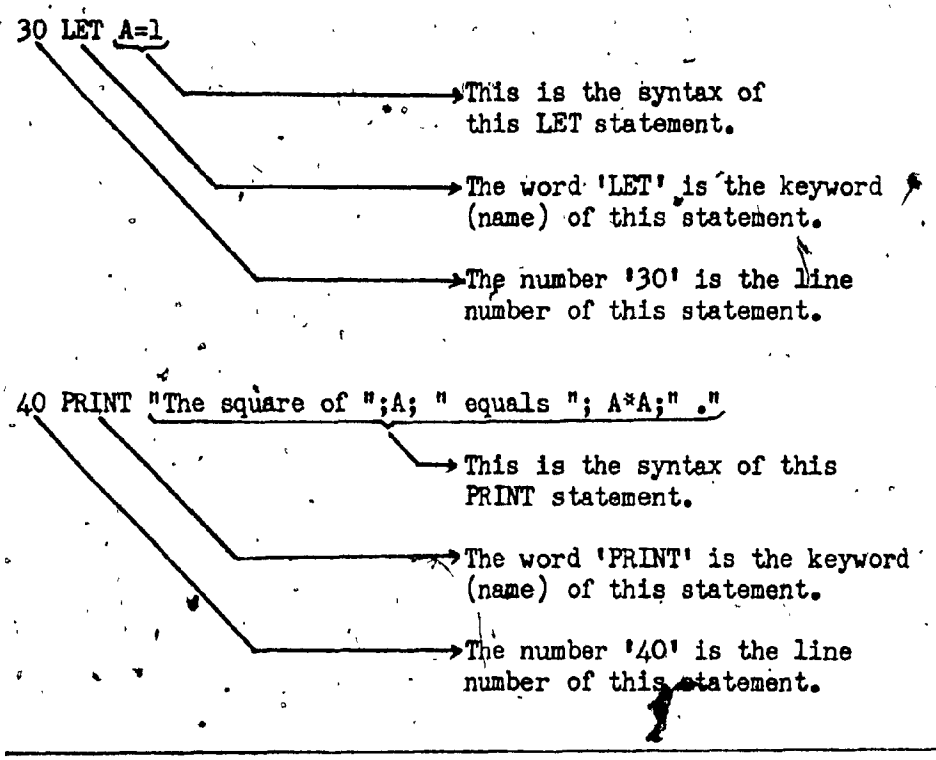
The number '10' is the line number of this statement.

```
20 REM squares of some numbers.
```

This is the syntax of this REM statement.

The word 'REM' is the keywrod (name) of this statement.

The number '20' is the line number of this statement.


30 LET A=1
This is the syntax of this LET statement.
The word 'LET' is the keyword (name) of this statement.
The number '30' is the line number of this statement.
40 PRINT "The square of ";A; " equals "; A*A;" ."
This is the syntax of this PRINT statement.
The word 'PRINT' is the keyword (name) of this statement.
The number '40' is the line number of this statement.

WHAT IS BASIC?

Introduction

BASIC (i.e. Beginner's All-Purpose Symbolic Instruction Code) is one of the computer programming languages. It was invented by John Kemeny and Thomas Kurtz in Dartmouth College, U.S.A. This programming language is applicable for general purposes especially for scientific and mathematical applications and computer-assisted instruction. The language is usually available on most minicomputers and many large computer systems and BASIC is the most common microcomputer language. (*Note: BASIC is not standardized and it can vary from one system to another. What is learned in this book sometimes cannot operate on some particular computer systems.)

What is it?

BASIC is an easy computer programming language which consists of 3 different kinds of characters:-

(1) Alphabetic characters:-

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

(2) Digits:-

1 2 3 4 5 6 7 8 9 0

(3) Special symbols:-

Name	Symbol
"Plus" sign	+
"Minus" sign	-
Asterisk (or "multiplication" sign)	*

Name	Symbol
Slash (or "division" sign)	/
Up arrow (or "exponentiation" sign)	↑
"Equals" sign	=
Point or period	.
Comma	,
Semicolon	;
Colon	:
Question mark	?
Left parenthesis	(
Right parenthesis	)
"Currency" symbol (Dollar sign)	$
"Less than" symbol	<
"Greater than" symbol	>
"Less than or equal to" symbol	=< or <=
"Greater than or equal to" symbol	=> or >=
"Not equal to" symbol	<>
"And" symbol	&
"Percentage" symbol	%
"Numeric" or character editing symbol	#
"At" symbol	@
Exclamation mark	!
Quotation marks (Double)	" "
Quotation mark (Single)	' '

All the above characters are similar to the English language. They are the bases to build the BASIC programming language. To solve a problem, a program is designed. When a program is designed in the BASIC programming language it is called a BASIC program

Example 1

Problem: If you want the computer to find the squares of some numbers (e.g. from 1 to 10), what should you do? You can design a program by using the BASIC programming language. The following is an example. (Note: It is not an example to teach how to calculate squares but to show how BASIC programming language is applied.)

Input program 1

```
10  REM  THIS IS A PROGRAM TO COMPUTE THE
20  REM  SQUARES OF SOME NUMBERS.
30  LET A=1
40  PRINT "THE SQUARE OF";A;" EQUALS"; A*A
50  LET A=A+1
60  IF A =<10 THEN 40
70  PRINT "JOB DONE."
80  END
```

This is a BASIC program that you feed into (type in) the computer terminal.

Input program 2

```
110  REM  THIS IS A PROGRAM TO COMPUTE THE
120  REM  SQUARES OF SOME NUMBERS.
130  LET A=1
140  PRINT "THE SQUARE OF";A;" EQUALS"; A*A
150  LET A=A+1
160  IF A=<10 THEN 140
170  PRINT "JOB DONE."
180  END.
```

This is also a BASIC program.

Program 1 and program 2 are the same except the number at the beginning of each line. You can start with any number you want. This number is called a line number in BASIC and no two lines have the same number. One line is one statement. Every time that you finish typing one line, hit the RETURN key on the keyboard. When the computer executes this program it will do it according to line sequence number. Let's look at each line closely:-

10 REM THIS IS A PROGRAM TO COMPUTE THE

This is a REM statement in BASIC programming language. It consists of 3 parts.

10 REM THIS IS A PROGRAM TO COMPUTE THE

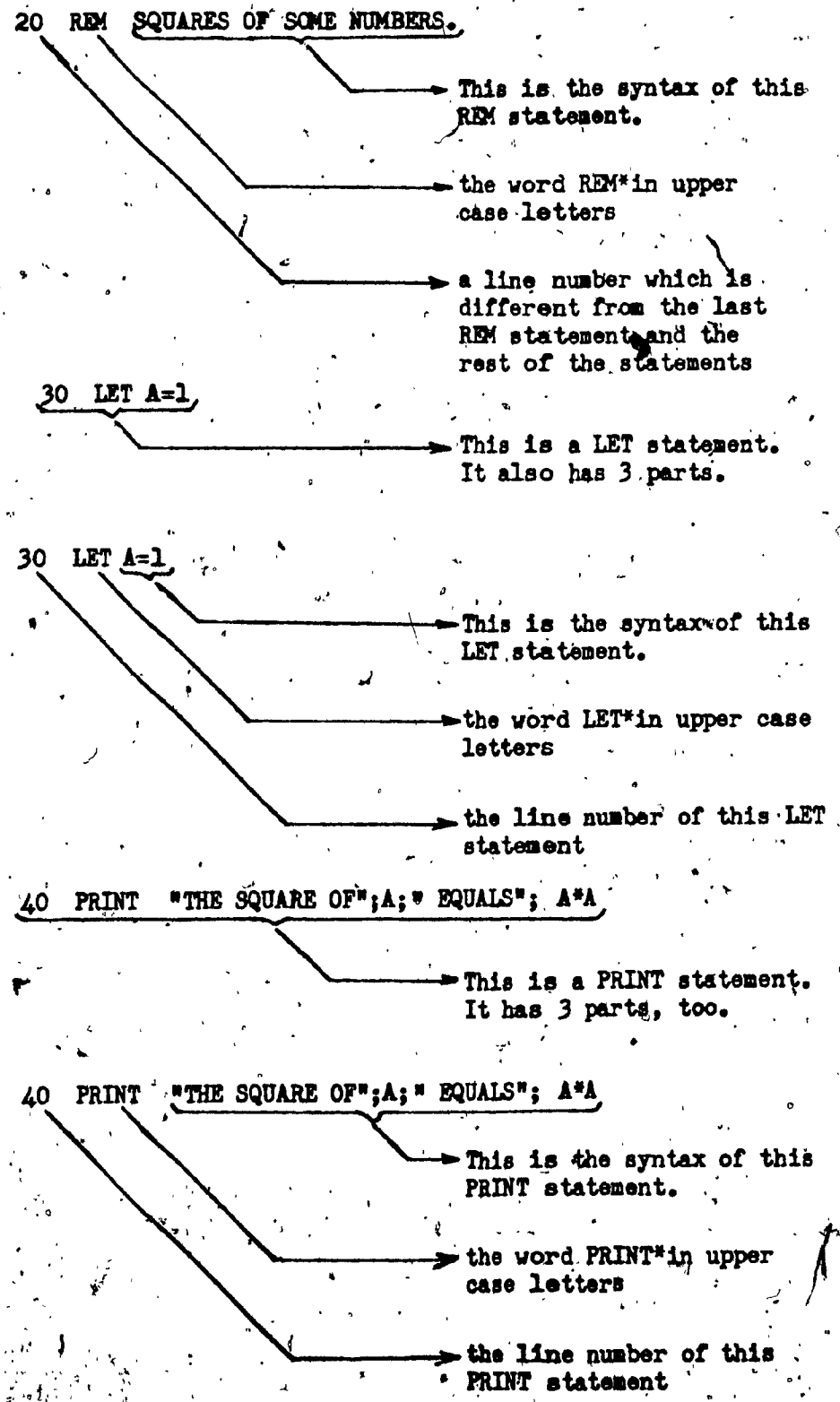
This is the syntax of this REM statement.

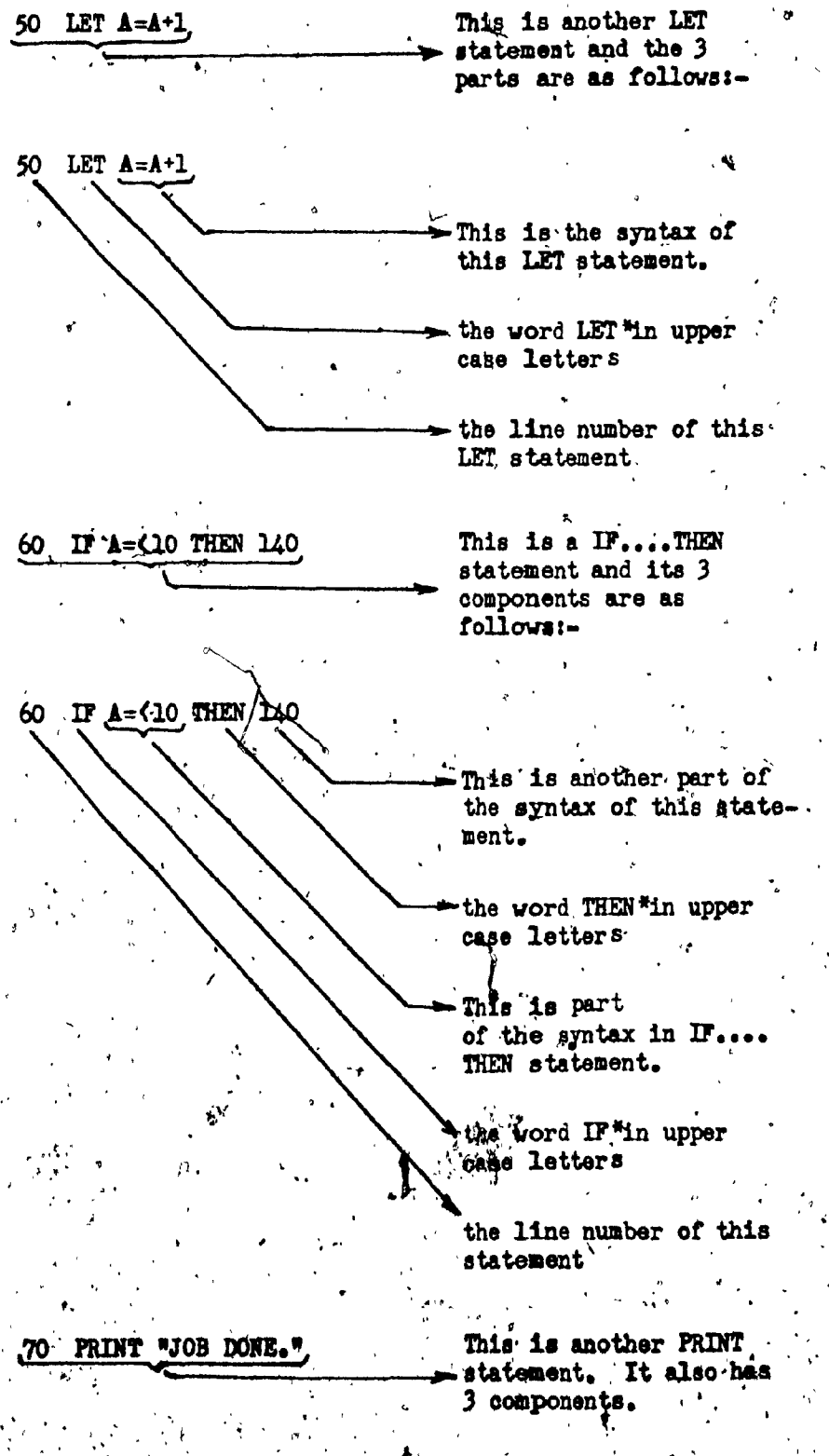
the word REM in upper case letters.(capital letters)

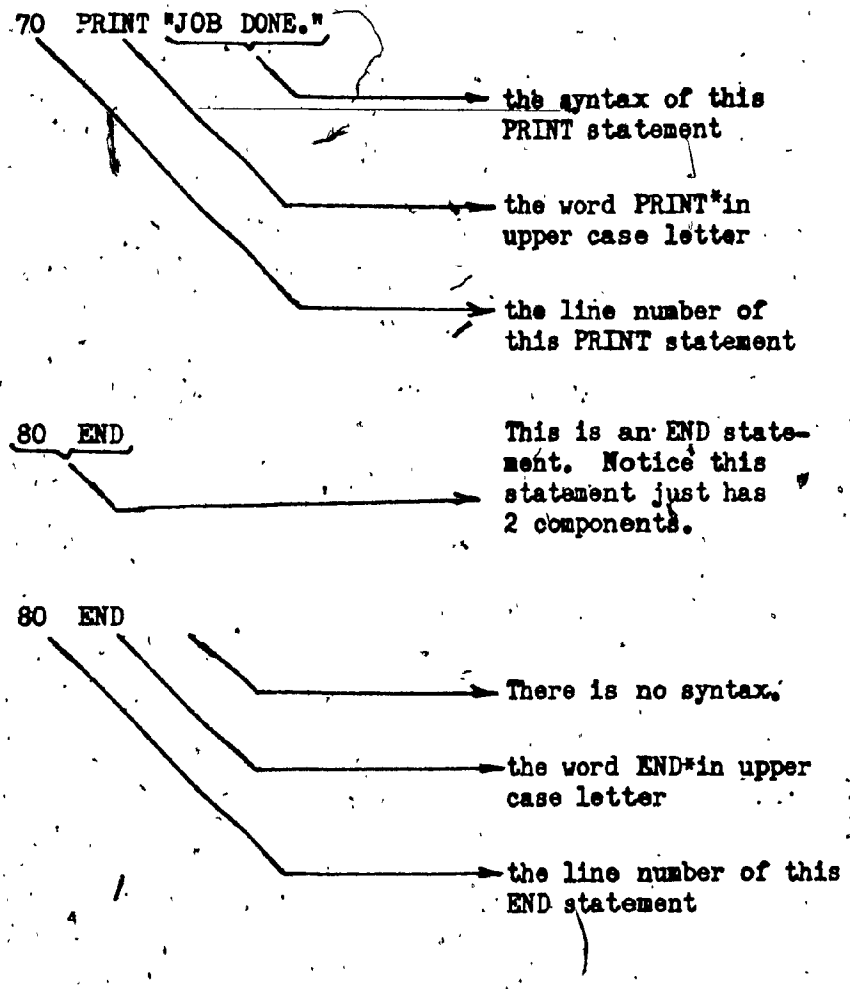
the line number of this REM statement

20 REM SQUARES OF SOME NUMBERS.

This is another REM statement. It also consists of 3 parts.


20 REM SQUARES OF SOME NUMBERS.
This is the syntax of this REM statement.
the word REM*in upper case letters
a line number which is different from the last REM statement and the rest of the statements
30 LET A=1
This is a LET statement. It also has 3 parts.
30 LET A=1
This is the syntax of this LET statement.
the word LET*in upper case letters
the line number of this LET statement
40 PRINT "THE SQUARE OF";A;" EQUALS"; A*A
This is a PRINT statement. It has 3 parts, too.
40 PRINT "THE SQUARE OF";A;" EQUALS"; A*A
This is the syntax of this PRINT statement.
the word PRINT*in upper case letters
the line number of this PRINT statement


50 LET A=A+1
This is another LET statement and the 3 parts are as follows:-
50 LET A=A+1
This is the syntax of this LET statement.
the word LET *in upper case letters
the line number of this LET statement
60 IF A=(10 THEN 140
This is a IF....THEN statement and its 3 components are as follows:-
60 IF A=(10 THEN 140
This is another part of the syntax of this statement.
the word THEN *in upper case letters
This is part of the syntax in IF.... THEN statement.
the word IF *in upper case letters
the line number of this statement
70 PRINT "JOB DONE."
This is another PRINT statement. It also has 3 components.



As seen from the above, a BASIC statement is an one-line instruction which consists of a line number, the name (keyword) of the statement and a syntax (except END and RETURN statements, a PRINT statement sometimes has no syntax). (See END, RETURN & PRINT statements for more information.)

*Note: In some computer systems, the keyword (name) of a statement doesn't have to be in upper case letters.

Output program

```
RUN
THE SQUARE OF 1 EQUALS 1
THE SQUARE OF 2 EQUALS 4
THE SQUARE OF 3 EQUALS 9
THE SQUARE OF 4 EQUALS 16
THE SQUARE OF 5 EQUALS 25
THE SQUARE OF 6 EQUALS 36
THE SQUARE OF 7 EQUALS 49
THE SQUARE OF 8 EQUALS 64
THE SQUARE OF 9 EQUALS 81
THE SQUARE OF 10 EQUALS 100
JOB DONE.
```

This is the PRINTOUT (the output or the result of your inputted program).

After you have typed in the input program and you want the computer to execute the program, you type RUN (RUN is a system command which asks the computer to carry out the operations in your program) and hit the RETURN key. After a short while the computer prints out the result for you (i.e. the above printout). The output is the desired outcome of the problem that you intended to solve with the help of the computer. When the computer is executing the statements in your input program, it is doing them in line by line sequence.

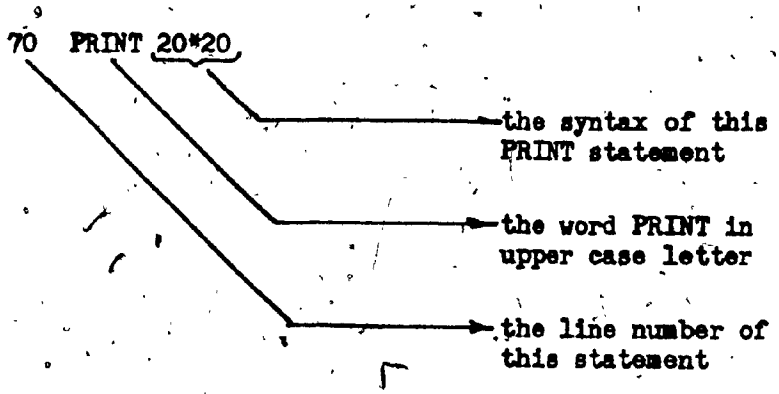
Example 2

Input program

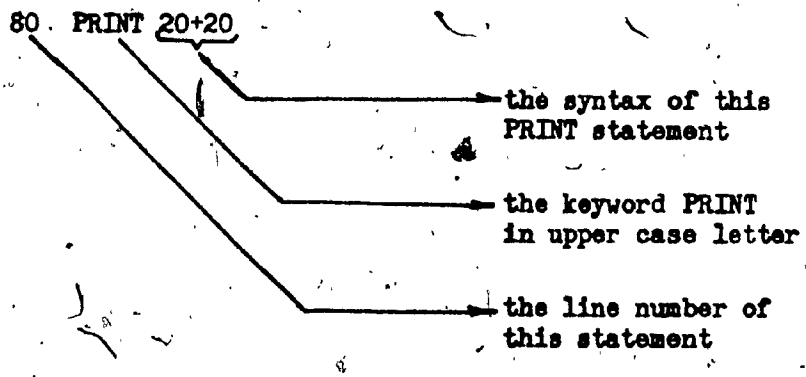
```
 70  PRINT 20*20
 80  PRINT 20+20
 90  PRINT 20-20
100  END
```

This is another BASIC program that has 4 statements.

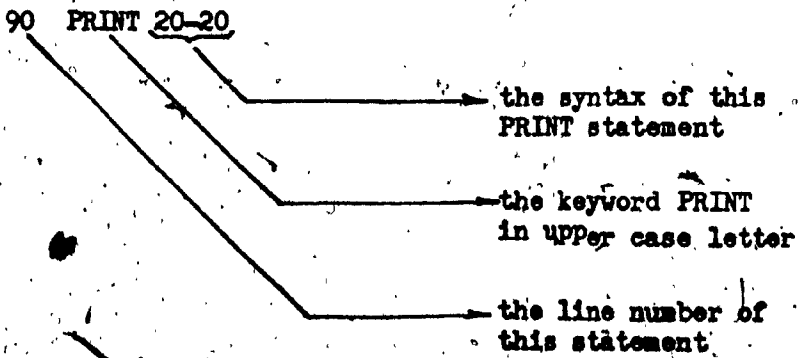
The first statement is a PRINT statement:-



The second statement is a PRINT statement, too:-



The third is also a PRINT statement:-



BASIC SYSTEM COMMANDS

NEW

OLD

RUN

LIST

REPLACE*
(RESAVE)

RESEQ

PURGE*
(UNSAVE,
SCRATCH,
CLEAR or
DESTROY)

CATALOG

BYE

SAVE

*The choice of word in system commands varies from one computer system to another. Some computer systems may have some other features.

Note: Only RUN & LIST system commands are enclosed for reference at the end of this course unit.

COMPONENTS AND SUB-COMPONENTS OF BASIC STATEMENTS

BASIC PROGRAMMING LANGUAGE

PROGRAM

A SET OF STATEMENTS

STATEMENT

- **Line number**
- **Name (keyword) of the statement**
  - REM
  - PRINT
  - PRINT TAB
  - PRINT USING
  - READ
  - DATA
  - LET
  - INPUT
  - GO TO
  - IF....THEN
  - ON....GO TO
  - FOR
  - NEXT
  - GOSUB , RETURN, DIM, MAT & END
- **Syntax of the statement (which may consists one or more of the followings:)**
  - **Constant (which consists of:)**
    - Numerical constant
    - String constant
    - Internal constant
  - **Variable (which consists of:)**
    - Numerical variable
    - String variable
  - **Operator (which consists of:)**
    - \+
    - \-
    - *
    - /
  - **Relational operator (which consists of:)**
    - =
    - <
    - >
    - =<
    - =>
    - <>
  - **Function (refer to the relevant pages concerned)**
  - **Punctuation (which consists of:)**
    - " "
    - ,
    - ;
    - :
    - $
    - etc.

COMPARE & CONTRAST TABLE ON:
SYSTEM COMMANDS & BASIC STATEMENTS

SYSTEM COMMAND	BASIC STATEMENT
(1) There is no line number.	(1) It must have a line number.
(2) It consists just only one word typed in upper case letters.	(2) It may consist more than one word also typed in upper case letters.
(3) There is no syntax.	(3) It must have a syntax. (except END & RETURN statements) (PRINT statement sometimes has no syntax.)
(4) It does not belong to BASIC programming language because some other programming languages also use some of these system commands.	(4) It is the main core of the BASIC programming language.
(5) It is not used to design programs but to direct the computer system to carry out a certain action. For example, SAVE, RUN, etc.	(5) It is used to design programmes. For example, 110 REM..... 120 PRINT..... 130 LET..... etc.

(6) Every system command has its own function.	(6) Every BASIC statement has its own function and characteristics and there are strict rules governing its own usage.

Related concepts: System command: RUN (p.176), LIST (p.179)

PART II: THE COMPONENTS OF A BASIC STATEMENT

SELF TEST

Note

This test determines how well you know the material in the coming map. If you can answer the following questions correctly, go to page 29. If not, go to page 26. When you are working on the questions, cover the answer section and check your results afterwards.

Exercises

Write down C(for correct) or I(for incorrect) for the followings:-

(1) Every BASIC statement must have a line number.
(2) No two BASIC statements can have the same line number.
(3) A line number is a number put in front of (to the left of) the keyword (the name) of a statement.

Answers

(1) C
(2) C
(3) C

LINE NUMBER

What is it?

A line number (or statement number) is a number headed in front of the name (keyword) of a statement. It is used to indicate the order of sequence (from a low number to a high number) in a BASIC program. Every BASIC program must have a line number. A line number could range from 0 to 99999 (*Note: Some systems do not allow 99999 as the highest line number.) and you can start with any number you like. However, no two statements can have the same line number. The statements are stored and executed by the computer in the order of their line numbers.

Example 1

```
110 LET A=20
111 LET B=40
112 PRINT A
113 PRINT B
```

Example 2

```
110 LET A=20
120 LET B=40
130 PRINT A
140 PRINT B
```

Explanation

The programs of the above two examples are the same except for their line numbers. In the first example there is no line number room between each line but in the second example there are 9 line number room between each statement. It is a good programming practice to have 9 spaces in line number between each statement.

so that you can later insert statements in between if necessary. However, if all the statements are arranged like example 1, you will have no line number space to insert any statement between lines when needed.

Note

When you are typing (keying) your program on the terminal you don't have to type your statements according to line number sequence. That is to say, you don't have to type statement 10 first, statement 20 second, statement 30 third, etc. Even if you type statement 30 first, statement 10 second, and statement 20 third, it doesn't bother the computer. When you ask the computer to RUN or LIST your program, it will arrange your statements in sequential order.

Example 3

Input program

```
130  LET A=A+1
```

→ You type line 130 first.

```
110  LET A=1
```

→ You type line 110 second.

```
120  PRINT "THE SQUARE OF ";A; "EQUALS " ; A*A
```

→ Lastly you type line 120.

```
LIST
110 LET A=1
120 PRINT "THE SQUARE OF "; A;"EQUALS" ; A*A
130 LET A=A+1
```

Explanation

The input program was not typed in according to line number sequence. However, after you have typed the word LIST in upper case letter and hit the RETURN key, the computer prints out the statements in line sequential order as shown above.

Rule

A line number is needed for every BASIC statement.

Exercises

Write down T (for true) or F (for false) for the following :-

(1) Every BASIC statement must be headed by a line number.
(2) We have to type in our statements according to line number sequence.

Answers

(1) T
(2) F

Related concepts

System command: LIST (p.179)

SELF TEST

Note	The following test determines how well you know the material in the coming map. If you can answer the question correctly, go to page 32. If not, go to next page. When you are working on the question, cover the answer section and check your result afterwards.
Exercise	List 5 names (keywords) of statements in BASIC.
Answer	REM, LET, PRINT, INPUT, DATA, etc. (or equivalent answer)

THE NAME (KEYWORD) OF A STATEMENT

What is it? The name (keyword) of a BASIC statement refers to the words placed after (to the right of) the line number in a BASIC statement. The name of any statement has to be typed in UPPER CASE LETTER (capital letter).* (*Note: In some systems it is not necessary to type the keyword in upper case letters.) It may consist of more than one word.

Examples

```
110 REM  THIS IS A PROGRAM TO COMPUTE
120 REM  THE SQUARES OF SOME NUMBERS.
130 LET A=1
140 PRINT "THE SQUARE OF "; A;"EQUALS "; A*A
```

Explanation

```
110 REM  THIS IS A PROGRAM TO COMPUTE
```

→ the keyword REM typed in upper case letter

```
120 REM  THE SQUARES OF SOME NUMBERS.
```

→ The same keyword REM is typed in upper case letter.

```
130 LET A=1
```

→ the word LET typed in upper case letter

```
140 PRINT "THE SQUARE OF ";A; "EQUALS "; A*A
```

→ the word PRINT typed in upper case letter

Some more examples are: READ, DATA, INPUT, LET, GO TO, IF....THEN, etc.

Non-examples RUN, LIST, SAVE, OLD, BYE, NEW, etc.

All these are system commands and they are not used to design programs. They are used to direct the computer system to carry out certain actions.

SELF TEST

Note

If you can answer the following questions correctly, go to page 37. If not, turn to next page.

Exercises

(1) What is the syntax of a BASIC statement?
(2) Name the 3 kinds of constants used in BASIC.
(3) Name the 2 kinds of variables used in BASIC.
(4) Name 2 other kinds of syntax (aprt from constant and variable) used in BASIC statements.

Answers

(1) The syntax of a BASIC statement is the written format put after (to the right of) the name (keyword) of a BASIC statement.
(2) (a) numerical constant,
(b) string constant, and
(c) internal constant.
(3) (a) numerical variable, and
(b) string variable.
(4) operator, relational operator (or function and punctuation mark)

OVERVIEW OF THE STATEMENT SYNTAX

What is it?

The syntax is the written format put after (to the right of) the name (keyword) of a BASIC statement. The syntax may contain one or more of the following commponents:-

(A) Constant which consists of 3 kinds:-
  (1) Numerical constant,
  (2) String constant, and
  (3) Internal constant.
(B) Variable which consists of 2 kinds:-
  (1) Numerical variable, and
  (2) String variable.
(C) Operator,
(D) Relational operator,
(E) Function,
(F) Punctuation.

There are fixed rules for the syntax in every BASIC statement and you have to comply with them in designing your program or the computer will give you a syntax error.

Examples

```
20 PRINT "PLEASE TYPE YOUR NAME."
30 INPUT X$
40 PRINT X$;", IT'S NICE TO MEET YOU."
```

Explanation

```
20  PRINT "PLEASE TYPE YOUR NAME."
```

This is the syntax of this PRINT statement and it is called a string constant.

```
30  INPUT X$
```

This is the syntax of this INPUT statement and it is called a string variable.

```
40  PRINT X$;", IT'S NICE TO MEET YOU."
```

The syntax of this PRINT statement has 2 parts. This part is called a string constant.

This is another part and it is called a string variable.

Further discussion of the syntax of BASIC statement is found in the following pages.

Note

END and RETURN statements do not have any syntax. (See END and RETURN statements). PRINT statement can have no syntax (see PRINT statement.)

Related concepts

Constant (p.35), numerical constant (p.38), string constant (p.48) Internal constant (p.54), Variable (p.56), Numerical variable (p.60), String Variable (p.66), Operator (p.74), Relational Operator (p.81), Punctuation Mark (p.84)

STATEMENT SYNTAX: CONSTANT

Definition

A constant is the data item which has a fixed (unchanging) value. That is to say the value of a constant does not change during the execution of the program. A constant can be a number or a text.

Kinds of constant

There are 3 kinds of constant:-

(A) Numerical constant,
(B) String constant, and
(C) Internal constant.

Example 1

```
70 LET A=10
```

This is the syntax of this LET statement.

```
70 LET A=10
```

10 is a numerical constant. Notice only 10 in this statement is a constant and nothing else.

Example 2

```
130 PRINT "GOOD MORNING!"
```

This is the syntax of this PRINT statement and it is called a string constant.

Related concepts	Numerical constant (p.38), String constant (p.48), Internal constant (p.54)

SELF TEST

Note — If you can answer the following questions correctly, go to page 47. If not, go to next page.

Exercises

(1) Study the following statements:-

20 LET A=50

30 LET B=-0.559

What are '50' and '-0.559' called in the above statements?

(2) What is the following called in BASIC:-

1.23E+2

(3) What is a floating point notation used for in BASIC?

(4) How can you change 5.1798E+10 into its ordinary notation?

Answers

(1) They are numerical constants.

(2) a floating point notation

(3) A floating point notation is used by computer to express:-

(a) a very large number, or

(b) a very small decimal fraction.

(4) See page 42.

STATEMENT SYNTAX: NUMERICAL CONSTANT

Definition

A numerical constant is a kind of constant used in BASIC. It refers to any number in all forms. In BASIC, a numerical constant can be written as a whole number, a decimal number, negative or positive, or expressed in floating point notation.

Examples 1

```
20 LET A=50
```

→ This is the syntax of this LET statement.

```
20 LET A=50
```

→ '50' is a numerical constant.

```
30 LET B=-5.59
```

→ This is the syntax of this LET statement.

```
30 LET B=-5,59
```

→ But, only '-5.59' is a numerical constant.

More examples for numerical constant are:-

134 → This is a positive numerical constant.

-564	This is a negative numerical constant.
0.97	This is a positive decimal numerical constant.
-5.57	This is a negative decimal numerical constant.
1.23E+2	This is a numerical constant expressed in floating point notation Note there is a letter E in this constant. It is called an exponent sign. The value to the left of E is multiplied by 10 raised to the power of the integer following the E.

Note

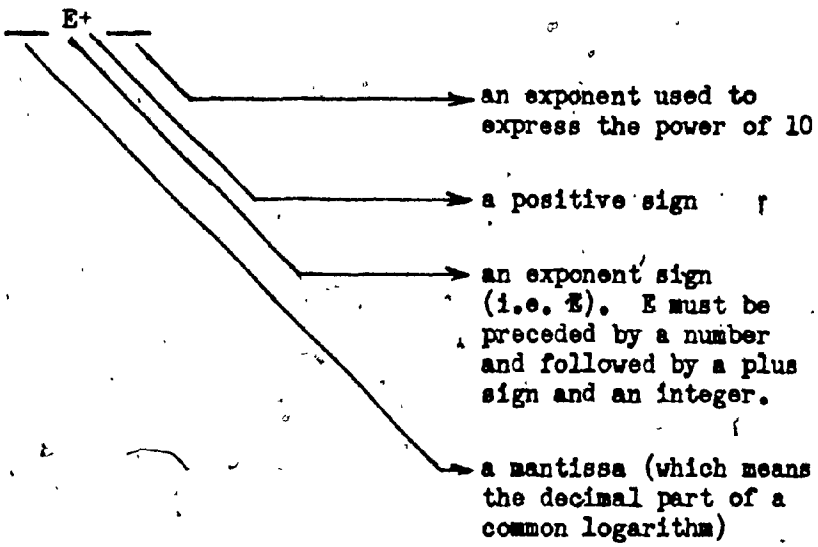
Floating point notation is commonly used in BASIC. It is called a floating point notation because the decimal point "floats" from one place to another when it is changed into its ordinary notation. A floating point notation is a shorthand for computer to express:-

(1) a very large number, or

(2) a very small decimal fraction.

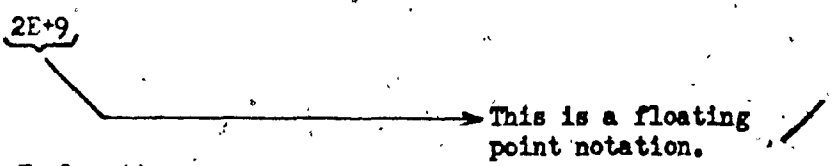
Examples 2 <u>To express a very large number by floating point notation</u>

Its format is:-

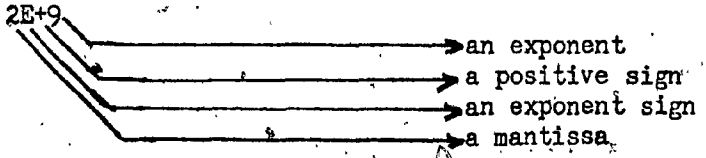


*Note: The two horizontal lines shown in the format above are for illustrative purposes. They do not appear in the BASIC program.

<u>Example</u>



<u>Explanation</u>



2E+9 means 2 x 10^9 (i.e. 2,000,000,000).

The way to convert 2E+9 to ordinary notation is:-

(1) Write down 2

(2) Write down 9 zero & add the decimal point after the 9th zero.

(3) Add commas.

Then it will become 2,000,000,000.

Example

5.1798E+10

→ a floating point notation

Explanation

5.1798E+10

→ an exponent

→ a positive sign

→ an exponent sign

→ a mantissa

5.1798E+10 means 5.1798 x 10^{10} (i.e. 51,798,000,000).

The way to convert 5.1798E+10 to ordinary notation is:-

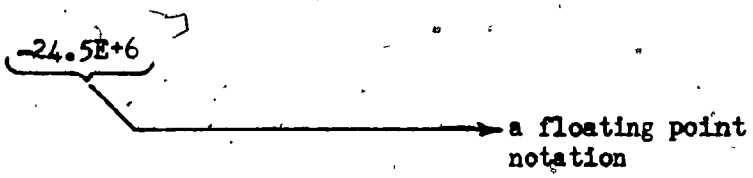
(1) Write down the number 51798.

(2) Count 10 decimal points after 5.

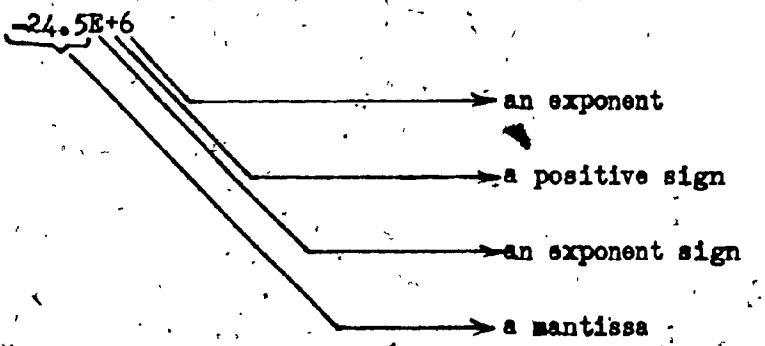
(3) Write down 6 zeros after 8 and add commas.

Then, it will become 51,798,000,000.

Example



Explanation



-24.5E+6 means -24.5×10^{6}.

That is -24,500,000.

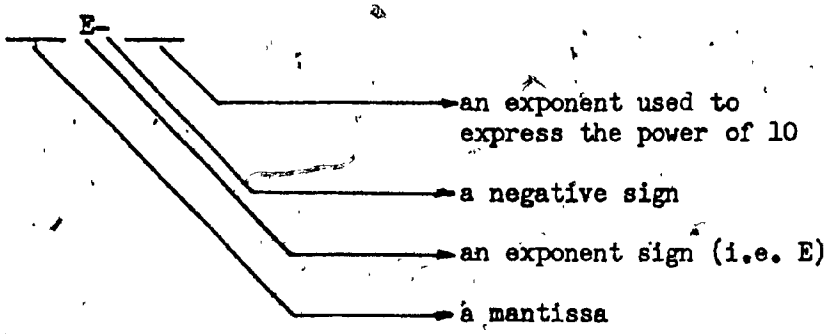
The way to convert -24.5E+6 to ordinary notation is:-

(1) Write down the number -245.
(2) Count 6 decimal points after 4.
(3) Add 5 zeros after 5 and add commas.

Then, the result will be -24,500,000.

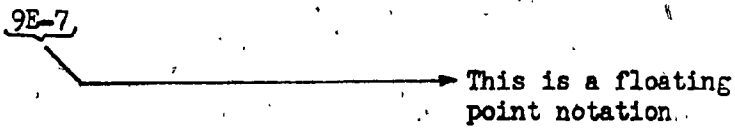
Examples 3 **To express a very small decimal fraction by floating point notation**

Its format is:-



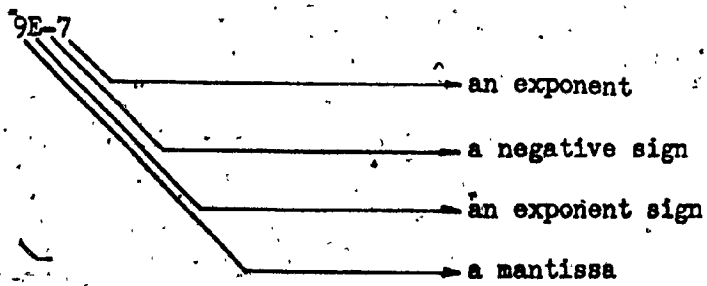
*Note: The two horizontal lines shown on either side of 'E-' are for illustrative purposes. They do not appear in your BASIC program.

Example



*Note: In the example above the number after 'E-' is an exponent and shows how many places to shift the decimal point to the left to change to its ordinary notation.

Explanation



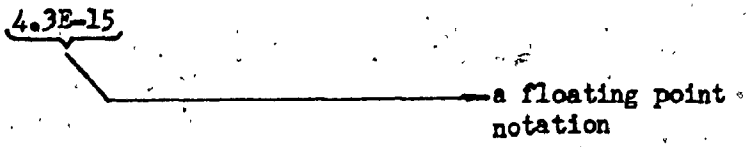
9E-7 means 9×10^{-7} (i.e. 0.0000009).

The way to convert 9E-7 to ordinary notation is:-

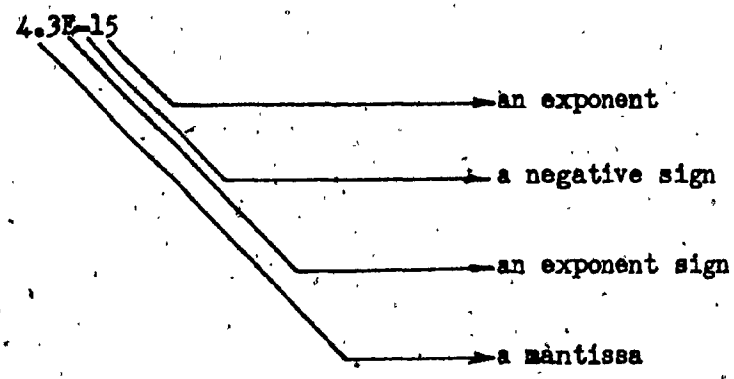
(1) Write down the number 9
(2) Add 6 zeros to the left of 9.
(3) Put down a dot to the left of the 6th zero to the left of 9.

Then, it will become 0.0000009.

Example



Explanation



4.3E-15 means 4.3×10^{-15} (i.e. 0.0000000000000043).

The way to convert 4.3E-15 to ordinary notation is:-

(1) Write down the number 43.
(2) Count 15 decimal points to the left of 3.
(3) Add 14 zeros to the left of 4.
(4) Put down a dot to the left of the 14th zero to the left of 4.

Then, it will become 0.00000000000043.

The following table show the equivalance of exponential notation and ordinary notation:-

Exponential Notation	Ordinary Notation
10^{1}	10
10^{2}	100
10^{3}	1,000
10^{4}	10,000
10^{5}	100,000
10^{6}	1,000,000
10^{7}	10,000,000
10^{8}	100,000,000
10^{9}	1,000,000,000
10^{10}	10,000,000,000
10^{0}	1
10^{-1}	0.1
10^{-2}	0.01
10^{-3}	0.001
10^{-4}	0.0001
10^{-5}	0.00001
10^{-6}	0.000001
10^{-7}	0.0000001
10^{-8}	0.00000001
10^{-9}	0.000000001
10^{-10}	0.0000000001

Exercises

Write down T (for true) and F (for false) for the followings:-

(1) A numerical constant can be any number.
(2) A floating point notation can be a numerical constant.
(3) A floating point notation can be a numerical constant used by a computer to express a very large number or a very small decimal fraction.
(4) This is a floating point notation: 5.7E-10
(5) This is a floating point notation: 7E+9

Answers

(1) T
(2) T
(3) T
(4) T
(5) T

Related concepts

Constant (p.35), string constant (p.48), Internal constant (p.54)

SELF TEST

Note

If you can answer the following questions correctly, go to page 54. If not, go to page 48.

Exercises

(1) What is a string constant?

(2) Study the following statements:-

```
10 PRINT "XYZ"
20 LET A=567
30 LET A$="567"
40 PRINT "567"
```

(a) What is 'XYZ' called in BASIC (line 10)?

(b) What is '567' called in BASIC (line 20)?

(c) What is '567' called in BASIC (line 30)?

(d) What is '567' called in BASIC (line 40)?

Answers

(1) A string constant is a text which is only found inside the quotation marks.

(2) (a) 'XYZ' is a string constant.

(b) '567' in line 20 is a numerical constant.

(c) '567' in line 30 is a string constant.

(d) '567' in line 40 is a string constant.

STATEMENT SYNTAX: STRING CONSTANT

Definition

A string constant is a kind of constant. It is a text or anything which is only found inside the quotation marks (i.e. opening and closing quotation marks) in PRINT, LET or DATA statements. The opening quotation marks tell the computer where the string constant starts, and the closing quotation marks tell it where the string constant ends. The string constant includes everything inside the opening and the closing quotation marks e.g. words, numbers, symbols, etc and even spaces.

Example 1

```
 90  PRINT "HELLO!"
100  PRINT "MY NAME IS JOE."
110  PRINT " I AM 25 years OLD."
120  PRINT "HOW ARE YOU?"
```

These are the statements you type on the terminal.

Explanation

```
90  PRINT "HELLO!"
```

→ This is a string constant in this PRINT statement.

```
90  PRINT "HELLO!"
```

→ ! is the end of the string constant.

→ The letter 'H' is the beginning of the string constant.

```
100 PRINT "MY NAME IS JOE."
```

The whole sentence inside the quotation marks is a string constant including the period at the end of the sentence.

```
100 PRINT "MY NAME IS JOE."
```

The period is the end of the string constant.

The letter 'M' is the beginning of the string constant.

```
110 PRINT " I AM 25 years OLD."
```

This is the string constant in this PRINT statement.

```
110 PRINT " I AM 25 years OLD."
```

The period here is the end of this string constant.

Note carefully here. The space right after the opening quotation marks is the beginning of this string constant.

```
120 PRINT "HOW ARE YOU?"
```

This question is a string constant.

```
120 PRINT "HOW ARE YOU?"
```

The question mark is the end of this string constant.

The letter H is the beginning of this string constant.

After having typed the 4 statements on the terminal, you type the word RUN in upper case letters and hit the RETURN key. The result of the output is:-

```
HELLO!
MY NAME IS JOE.
 I AM 25 years OLD.
HOW ARE YOU?
```

Explanation

The computer prints out exactly the same text inside the quotation marks in each PRINT statement, even a space in front of the sentence "I AM 25 years OLD.", and the word "years" is in small letters.*

Examples 2

```
10  LET A$="YES"
20  LET B$="NO"
30  LET C$="50"
40  PRINT A$
50  PRINT B$
60  PRINT C$
```

Explanation

```
10  LET A$="YES"
```

→ The word 'YES' is a string constant.

*Note: In some systems (e.g. APPLE) there is no small letters (or lower case) while in some other systems (e.g. CDC) it must be in ASCII.

```
20 LET B$="NO"
```

The word 'NO' is a string constant.

```
30 LET C$="50"
```

The number '50' is a string constant.

```
40 PRINT A$
50 PRINT B$
60 PRINT C$
```

There is no string constant in all these PRINT statements.

Output program

```
RUN
YES
NO
50
```

All the text inside the quotation marks are printed out.

Example 3

```
10 READ A$
20 DATA "GOOD & BAD"
30 PRINT A$
```

Explanation

```
20 DATA "GOOD & BAD"
```

The text 'GOOD & BAD' is a string constant.

Output program

```
RUN
GOOD & BAD
```

After you type the word RUN and hit the RETURN key, the computer prints out the string constant.

Non-examples

```
40  PRINT "MY NAME IS JOE.
```

This is not a string constant because a closing quotation mark is missing.

```
40  PRINT MY NAME IS JOE."
```

This is a syntax error because an opening quotation mark is missing.

```
40  PRINT MY NAME IS JOE.
```

This is a syntax error, too because both opening and closing quotation marks are missing.

```
50  PRINT 'MY NAME IS JOE.'
```

This is a syntax error again because the wrong kind of quotation marks are used. (*Note: Some computer systems accept this kind of quotation marks.)

Rule

Both opening and closing quotation marks are needed to enclose a string constant.

Exercises

Write down T (for true) or F (for false) for the followings:-

(1) 250 PRINT "%*&"
In the above statement, '%*&' is a string constant.

(2) 50 PRINT "567"
In the above statement, '567' is a numerical constant.

(3) 50 LET A=567
In the above statement, '567' is a numerical constant.

(4) 60 LET A$="567"
In the above statement, '567' is a numerical constant.

Answers

(1) T

(2) F (*Note: '567' is a string constant because it is enclosed in quotation marks in this PRINT statement.)

(3) T

(4) F (*Note: '567' is a string constant because it is enclosed in quotation marks in this LET statement.)

Related concepts

Statement syntax (p.33), LET statement (p.123), PRINT statement (p.95), variable (p.56)

STATEMENT SYNTAX: INTERNAL CONSTANT

What is it?

An internal constant is the value that is fixed or predefined inside the computer.

Examples

The following values are fixed in the computer:-

(1) natural log
(2) pi
(3) square of 2
(4) centimeters per inch
(5) kilograms per pound
(6) liters per gallon

The following is the table of exchange of values:-

Constant	BASIC symbol	Value
e (natural log)	&B	2.718281828459
π (pi)	&PI	3.141592653590
$\sqrt{2}$ (square root of 2)	&SQR2	1.414213562373
centimeters per inch	&INCM	2.54
kilograms per pound	&LBKG	0.453592
liters per gallon	&GALI	3.785412

Related concepts

Statement syntax (p.33), Numerical constant (p.38), string constant (p.48), Variable (p.56)

SELF TEST

Note

If you can answer the following questions correctly, go to page 59. If not, go to next page.

Exercises

Identify all the variables used in the following statements:-

```
70 LET A=10
80 LET B=20
90 LET A$="GOOD"
100 LET B$="BAD"
```

Answers

'A' in line 70.
'B' in line 80.
'A$' in line 90.
'B$' in line 100.

STATEMENT SYNTAX: VARIABLE

Definition

A variable is a label or sign which is used to be assigned with a value. A variable has a place in the memory of the computer where its value is stored.

Kinds of variable

There are 2 kinds of variables:-

(1) Numerical variable (which is used to be assigned with a value from a numerical constant), and
(2) String variable (which is used to be assigned with a value from a string constant).

Examples 1

```
70 LET A=10
80 LET B=20
```

Explanation

```
70 LET A=10
```

→ The letter 'A' is a numerical variable. Note '10' is a numerical constant. The numerical variable is assigned the value '10'.

```
80 LET B=20
```

→ The letter 'B' is a numerical variable and it is assigned the value '10'.

Examples 2

```
50 LET A$="GOOD"
60 LET B$="BAD"
70 PRINT A$
80 PRINT B$
```

Explanation

```
50 LET A$="GOOD"
```

→ The label 'A$' is a string variable and the word 'GOOD' is a string constant (Still remember?). A$ is assigned with the value of 'GOOD'.

```
60 LET B$="BAD"
```

→ The label 'B$' is a string variable. The word 'BAD' is a string constant. 'B$'is assigned with the value of 'BAD'.

```
70 PRINT A$
```

→ This statement asks the computer to print out the value of 'A$'.(Note: This does not ask the computer to print 'A$'.)

```
80 PRINT B$
```

→ The label 'B$' is the same as in line 60. This statement asks the computer to print out the value of the string variable 'B$'.

Exercises

Write out the variables used in the following statements:-

```
10  LET H$="HIGH"
20  LET L$="LOW"
30  LET Z=700
```

Answers

Line 10: H$
Line 20: L$
Line 30: Z

Related concepts

Variable (p.56), String variable (p.66), LET statement (p.123), PRINT statement (p.95), Compare & Contrast table on: Numerical & string variables (p.71)

SELF TEST

Note

If you can answer the following questions correctly, go to page 65. If not, go to next page.

Exercises

(1) Identify the numerical variables used in the following statements:-

```
10 LET A=5
20 LET X4=20
30 LET V7=70
```

(2) Debug the following statements:-

```
20 LET *K=20
30 LET Y$=80
40 LET X#=50
```

(3) How is a numerical variable formed in BASIC?

Answers

(1) 'A' in line 10.
'X4' in line 20.
'V7' in line 30.

(2) Revised statements

```
20 LET K=20
30 LET Y=80
40 LET X=50
```

(3) A numerical variable can be formed by:-
(a) any alphbet (e.g. A,B,C....or Z), or
(b) any alphbet (i.e. A,B,C....or Z) plus one digit (i.e. 0,1,2,3....or 9)

STATEMENT SYNTAX: NUMERICAL VARIABLE

Definition

A numerical variable is a variable which is used to be assigned with a value from a numerical constant. There are 286 places (or spaces) inside the computer memory where numerical variables are stored. You can imagine that there are 286 boxes in the computer memory (shown below). In other words, BASIC only allows these 286 numerical variables for you to design in your program.

A	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
G	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9
H	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9
I	I0	I1	I2	I3	I4	I5	I6	I7	I8	I9
J	J0	J1	J2	J3	J4	J5	J6	J7	J8	J9
K	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9
L	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9
M	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
N	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9
O	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9
P	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
Q	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9

R	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
S	S0	S1	S_2	S3	S4	S5	S6	S7	S8	S9
T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
U	U_0	U1	U2	U3	U4	U5	U6	U7	U8	U9
V	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
W	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9
X	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
Y	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
Z	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9

286 Numerical variables in BASIC

Rules

The rules to form a numerical variable are:-

(1) any letter e.g. A,B,C....or Z

or,

(2) any letter e.g. A,B,C...or Z plus one digit only e.g. 0,1,2....or 9

Notes

The digit zero (0) and the letter O are very confusing. Try not to use them as numerical variables in programming. Numerical variables are only used to be assigned to numerical values (i.e. numerical constants).

Examples

```
10 LET A=5
20 LET X3=100
30 LET V8=27
```

Explanation

```
10 LET A=5
```

→ The label 'A' is a numerical variable. It is assigned the value of '5' (i.e. it takes the value from a numerical constant.)

```
20 LET X3=100
```

→ The label 'X3' is a numerical variable. It is assigned the value of '100' (a numerical constant).

```
30 LET V8=27
```

→ The label 'V8' is a numerical variable. It is assigned the value of '27' (a numerical constant).

Non-examples

```
10 LET AA=5
```

→ The label 'AA' is not a numerical variable because two letters are not allowed.

```
20 LET Z77=100
```

→ The label 'Z77' is not a numerical variable because two digits are not allowed.

30 LET R#=27

The label 'R#' is not a numerical variable because this symbol '#' is not allowed.

40 LET Z;=50

The label 'Z;' is not a numerical variable because a punctuation mark is not allowed.

The following are not allowed as numerical variables -

(1) Too many letters:-

e.g. AS, HU, EM, MKL, etc.

(2) Wrong placement:-

e.g. 8H, 9Y, 5K, 2M, etc.

(3) Digits:-

e.g. 8, 56, 89, 109, etc.

(4) Punctuation marks and special signs

H*, K+, $, #, etc.

Rule

In other words, apart from the 286 numerical variables specified, no other form of numerical variable is allowed to be used as a numerical variable.

Note

The statements that frequently use numerical variables are:-

PRINT, LET, INPUT, IF....THEN, READ,
ON....GO TO, FOR & NEXT, etc.

Exercises

Write down C (for correct) or I (for incorrect) for the following statements:-

(1) 50 LET 5W=50
(2) 70 LET Z=80
(3) 20 LET K*K=270
(4) 40 LET M9=9
(5) 20 LET B=50
    30 LET B=B+1

Answers

(1) I (because '5' cannot be put in front of 'W' in '5W')
(2) C
(3) I (because '*' is not accepted as a numerical variable in 'K*K')
(4) C
(5) C

Related concepts

Variable (p.56), String variable (p.66), LET statement (p.123), PRINT (p.95), Compare & contrast table on: numerical and string variables (p.71)

SELF TEST

Note

If you can answer the following questions correctly, go to page 73. If not, go to next page.

Exercises

(1) Identify the string variables in the following statements:-

```
10 LET A$="NAME"
20 LET B$="SEX"
30 LET A=50
40 LET B=80
```

(2) Debug the following statements:-

```
10 LET 7$="GOOD"
20 LET $A="SEX"
```

(3) How is a string variable formed in BASIC?

Answers

(1) 'A$' in line 10.
'B$' in line 20.

(2) Revised statements

```
10 LET B$="GOOD"
20 LET A$="SEX"
```

(3) A string variable is formed by:-

(a) an alphbet (i.e. A,B,C....or Z) plus a dollar sign (i.e. $), or

(b) an alphbet (i.e. A,B,C....or Z) plus a digit (i.e. 0,1,2,3....or 9) plus a dollar sign (i.e. $).

STATEMENT SYNTAX: STRING VARIABLE

Definition

A string variable is a variable which is used to be assigned with a value from a string constant. There are also 286 spaces inside the computer memory where string variables are stored. In other words, BASIC only allows these 286 string variables for you to design in your program.

A$	A0$	A1$	A2$	A3$	A4$	A5$	A6$	A7$	A8$	A9$
B$	B0$	B1$	B2$	B3$	B4$	B5$	B6$	B7$	B8$	B9$
C$	C0$	C1$	C2$	C3$	C4$	C5$	C6$	C7$	C8$	C9$
D$	D0$	D1$	D2$	D3$	D4$	D5$	D6$	D7$	D8$	D9$
E$	E0$	E1$	E2$	E3$	E4$	E5$	E6$	E7$	E8$	E9$
F$	F0$	F1$	F2$	F3$	F4$	F5$	F6$	F7$	F8$	F9$
G$	G0$	G1$	G2$	G3$	G4$	G5$	G6$	G7$	G8$	G9$
H$	H0$	H1$	H2$	H3$	H4$	H5$	H6$	H7$	H8$	H9$
I$	I0$	I1$	I2$	I3$	I4$	I5$	I6$	I7$	I8$	I9$
J$	J0$	J1$	J2$	J3$	J4$	J5$	J6$	J7$	J8$	J9$
K$	K0$	K1$	K2$	K3$	K4$	K5$	K6$	K7$	K8$	K9$
L$	L0$	L1$	L2$	L3$	L4$	L5$	L6$	L7$	L8$	L9$
M$	M0$	M1$	M2$	M3$	M4$	M5$	M6$	M7$	M8$	M9$
N$	N0$	N1$	N2$	N3$	N4$	N5$	N6$	N7$	N8$	N9$
O$	O0$	O1$	O2$	O3$	O4$	O5$	O6$	O7$	O8$	O9$
P$	P0$	P1$	P2$	P3$	P4$	P5$	P6$	P7$	P8$	P9$

Q$	Q0$	Q1$	Q2$	Q3$	Q4$	Q5$	Q6$	Q7$	Q8$	Q9$
R$	R0$	R1$	R2$	R3$	R4$	R5$	R6$	R7$	R8$	R9$
S$	S0$	S1$	S2$	S3$	S4$	S5$	S6$	S7$	S8$	S9$
T$	T0$	T1$	T2$	T3$	T4$	T5$	T6$	T7$	T8$	T9$
U$	U0$	U1$	U2$	U3$	U4$	U5$	U6$	U7$	U8$	U9$
V$	V0$	V1$	V2$	V3$	V4$	V5$	V6$	V7$	V8$	V9$
W$	W0$	W1$	W2$	W3$	W4$	W5$	W6$	W7$	W8$	W9$
X$	X0$	X1$	X2$	X3$	X4$	X5$	X6$	X7$	X8$	X9$
Y$	Y0$	Y1$	Y2$	Y3$	Y4$	Y5$	Y6$	Y7$	Y8$	Y9$
Z$	Z0$	Z1$	Z2$	Z3$	Z4$	Z5$	Z6$	Z7$	Z8$	Z9$

286 String Variables in BASIC

Rules

The rules to form a string variable are:-

(1) a letter (i.e. A,B,C....or Z) plus a dollar sign (i.e. $),

or

(2) a letter (i.e. A,B,C....or Z) plus a digit (i.e. 0,1,2,3....9) plus a dollar sign (i.e. $)

Note

The digit zero (0) and the letter O are very confusing. Try not to use them to form string variables in your program.

Examples 1

```
10 LET A$="GOOD"
20 LET Z$="BAD"
```

Explanation

```
10 LET A$="GOOD"
```

The label 'A$' is a string variable. It is assigned the value of 'GOOD' (i.e. it takes the value from a string constant).

```
20 LET Z$="BAD"
```

The label 'Z$' is a string variable. It is assigned the value of 'BAD' (a string constant).

Non-examples 1

```
10 LET 2$="GOOD"
```

The label '2$' is not a string variable because a digit cannot be put alone before a dollar sign.

```
20 LET $A="GOOD"
```

The label '$A' is not a a string variable because a dollar sign cannot go before a letter.

```
30 LET A$B="GOOD"
```

The label 'A$B' is not a a string variable because the wrong order.

```
40  LET G="GOOD"
```

The letter 'G' is not a string variable. This is a syntax error. The letter 'G' is used as a numerical variable only.

Rules

In other words, apart from the 286 string variables mentioned in the previous pages, no other form of string variable is allowed to be used as a string variable. When you design your program make sure you use the correct string variables accepted by BASIC. Also make sure that you have assigned a string value to a string variable. In some computer systems there is a limit for the number of characters (or the length of text) to be assigned to a string variable. Some systems allow 4095 characters and others just allow 18 characters at its maximum.

Note

The statements that frequently use string variables are:-

PRINT, LET, INPUT, IF...THEN, READ, etc.

Exercises

Write down C (for correct) or I (for incorrect) for the followings:-

(1) 50 LET M="MOUTH"

(2) 80 LET $K="YES"

(3) 90 LET K$="NO"

Answers

(1) I (because the letter 'M' is used for numerical variable)

(2) I (because the dollar sign ($) cannot be put in front of a letter)

(3) C

Related concepts

Variables (p.56), Numerical variable (p.60), LET statement (p.123), PRINT statement (p.95), Compare & contrast table on: numerical & string variable (p.71)

COMPARE AND CONTRAST TABLE ON:

NUMERICAL AND STRING VARIABLES

The following is a table showing the similarities and the differences between numerical and string variables:-

NUMERICAL VARIABLE	STRING VARIABLE
(1) It is a kind of variable.	(1) It is a kind of variable.
(2) There are 286 numerical variables in BASIC.	(2) There are 286 string variables in BASIC.
(3) It is formed by:- (A) any letter e.g. A,B,C....or Z or, (B) any letter plus one digit only, e.g. A,B,C.... or Z + 0,1,2.... or 9.	(3) It is formed by:- (A) any letter plus a dollar sign ($) e.g. A,B,C....or Z + $ or, (B) any letter plus only one digit + a dollar sign, e.g. A,B,C.... or Z + 0,1,2.. or 9 + $

(4) It is used to be assigned a numerical value.	(4) It is used to be assigned a string value.
(5) The statements that frequently use numerical variables are:- PRINT, LET, INPUT, IF....THEN, READ, ON....GO TO, FOR & NEXT, etc.	(5) The statements that frequently use string variables are:- PRINT, LET, INPUT, IF....THEN, READ, etc.
(6) It cannot be used to be assigned a string value.	(6) It cannot be used to be assigned a numerical value.

SELF TEST

Note

You can answer the following questions correctly, go to page 80. If not, go to next page.

Exercises

(1) Write down the meanings of the followings:-

(a) 10/2

(b) 5*7

(c) 100↑2

(2) Name the priority of computation in the following arithmetic expression:-

SQR(25)+70-20/2↑2

Answers

(1) (a) 10 divided by 2

(b) 5 times 7

(c) 100 to the power 2

(2) See page 77.

STATEMENT SYNTAX: OPERATOR

What is it? An operator is an arithmetic sign which is used to imply an operation. There are a number of operators used in BASIC.

SIGN	MEANING
+	means addition
-	means substraction
*	means multiplication
/	means division
↑	means exponentiation* (raising to power)

Note They are very useful in expressing an arithmetic operation. When we use operators to express an arithmetic presentation it becomes an arithmetic expression.

Examples 1

```
10 PRINT 2+1
```

→ This statement asks the computer to print out the answer of 2 plus 1.

```
20 PRINT 2-1
```

→ This statement asks the computer to print out the answer of 2 minus 1.

*Note: Some systems use ** as an exponentiation symbol.

30 PRINT 2*7

This statement asks the computer to print out the answer of 2 times 7.

40 PRINT 10/2

This statement asks the computer to print out the answer of 10 divided by 2.

50 PRINT 2↑2

This statement asks the computer to print out the answer of 2 to the power 2.

Non-examples 1

10 PRINT 5+-2

It is wrong to use two arithmetic signs together.

20 PRINT 5↑-2

This is the same mistake again.

To get rid of ambiguity, you have to use parentheses. Remember that parentheses come in pairs. Every left parenthesis should be accompanied by a right parenthesis.

Rules

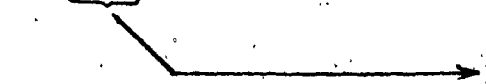
There is a fixed priority in doing computation by the computer. It always scans and works from the left to the right of the statement and the priority of computing expressions is as follows:-

(1) Firstly, the computer will do those within parentheses.
(2) Secondly, the computer will do functions.
(3) Thirdly, the computer will do exponentiation.
(4) Fourthly, the computer will do multiplication and division.
(5) Fifthly, the computer will do addition and subtraction.

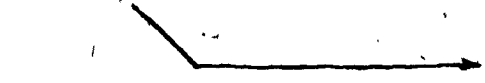
If there are two equal precedence (e.g. multiplication and division), the computer will perform the left most first in the expression. If you use more than one pair of parentheses, the inner-most pair of the parentheses will be computed first. The parentheses must go in pairs.

Example 2

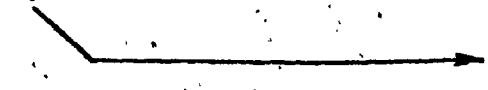
2*7+(5-2)-10↑2 — Priority of computation



2*7+3-10↑2



2*7+3-100



14+3-100

→ left to right rule applied here

17-100

→ lastly subtraction

-83

Example 3 SQR(25)+70-20/2↑2

→ function first

5+70-20/2↑2

→ then exponentiation

5+70-20/4

→ then division

5+70-5

→ left to right rule applied

75-5

→ lastly subtraction

70

(*Note: SQR means 'square root of'.)

Example 4

SQR(625)-SQR(6+(8/4*5))

→ right to left rule inside the inner most parentheses first, then the inner most parenthese

SQR(625)-SQR(6+10)

→ then parentheses

SQR(625)-SQR(16)

→ left to right rule applied

25-SQR(16)

→ then function

25-4

→ lastly subtraction

21

Notes

When you want to use arithmetic expressions in BASIC, bear the priority of computation in mind. If you misuse them the computer will give you a wrong answer. Special care must be taken in using exponentiation which means raising to the power.

Example

2*2 → means 2↑2 (two to the power 2)

Example

2*2*2*2 ——————— means 2↑4 (two to the power 4)

Exercises

(1) Write the meaning of the followings:-

(A) 10/2

(B) 5*7

(C) 100↑2

(2) Name the priority of computation in the following:-

SQR(81)-SQR(6+(8/4*5))

Answers

(1) (A) 10 divided by 2

(B) 5 times 7

(C) 100 to the power 2

(2) SQR(81)-SQR(6+(8/4*5))

→ 8 divided by 4 first

SQR(81)-SQR(6+(2*5))

→ then inner parentheses

SQR(81)-SQR(6+10)

→ then the parentheses

SQR(81)-SQR(16)

→ right to left rule applied

9-SQR(16)

→ then function

9-4

→ then subtraction

5

SELF TEST

Note

If you can answer the following questions correctly, go to page 83. If not, go to next page.

Exercises

(1) Explain the followings:-

(a) A=>B

(b) A<>B

(c) A<=B

(2) What is the function of a relational operator in BASIC?

Answers

(1) (a) A is equal to or greater than B.

(b) A is not equal to B.

(c) A is less than or equal to B.

(2) A relational operator is used to build up a relationship between 2 elements. Usually it is found in IF.... THEN statement for the purpose of comparison and decision. (or equivalent answer)

STATEMENT SYNTAX: RELATIONAL OPERATOR

What is it?

A relational operator is a sign (operator) which is used to build up a relationship between two elements. This kind of operator is not used to represent arithmetic expressions. The relational operators used in BASIC are:-

SIGN	MEANING
=	equal to
>	greater than
<	less than
<= or =<	less than or equal to
>= or =>	greater than or equal to
<>	not equal to

Examples

A>B ——→ It means A is greater than B.

A<=C ——→ It means A is less than or equal to C.

B<>M ——→ It means B is not equal to M.

Note

The above relational operators are extremely useful in the IF....THEN statement. They are used for the purpose of comparison and decision.

Exercises

Explain the followings:-

(1) X=>W

(2) X<>W

(3) X<=W

Answers

(1) X is equal to or greater than W.

(2) X is not equal to W.

(3) X is less than or equal to W.

SELF TEST

Note

If you can answer the following questions correctly, go to page 87. If not, go to next page.

Exercises

(1) Explain the meaning of commas and the semi-colon used in the following statement:-
10 PRINT A,B,C;D

(2) What is the use of quotation marks in a PRINT statement?

(3) What is the use of '$' in the following statement?
20 LET A$="NAME"

(4) What is the meaning of 'equal sign' (=) in the above statement in #(3)?

Answers

(1) The 1st comma means moving into the 1st tab zone. The 2nd comma means moving into the 2nd tab zone. The semi-colon means squeezing the printing space together.

(2) The quotation marks used in a PRINT statement are used to tell the computer where the string constant starts and ends. When the PRINT statement is executed, the computer will print out the string constant.

(3) The 'dollar sign' ($) tells the computer that it is a label to denote a string variable.

(4) The 'equal sign' (=) in the LET statement means 'replaced by'.

STATEMENT SYNTAX: PUNCTUATION MARKS

What is it? Most of the punctuation marks used in BASIC bear the same meaning as the English language. But some have special meanings in BASIC.

Examples

(1) Quotation marks (i.e. " ")

Quotation marks carry a similar meaning as our English language when they are used in BASIC. Quotation marks consist of an opening quotation mark and a closing quotation mark. They have to work in pairs. The opening quotation mark tells the computer where a string constant starts. The closing quotation mark tells the computer where a string constant ends. Quotation marks are particularly used in PRINT, LET, INPUT & DATA statements. (For more information please refer string constants.)

(2) Semicolon (i.e. ;) & comma (i.e. ,)

Both the semicolon and the comma have special meanings when they are used in the PRINT statement. They are to determine the printing format of the output.

(A) Semicolon (;) means squeezing the printing space together.

(B) Comma (,) means there are 5 tab (print) zones to move into (see page 88). There are 5 tab zones in one line and one tab zone consists of 15 printing spaces (except the last zone which has 12 printing spaces). The 1st tab zone is the 1st printing position. The 2nd tab zone is the 15th printing position. The 3rd tab zone is the 30th printing position. The 4th tab zone is the 45th printing position. The 5th tab zone is the 60th printing position. If you have commas in the PRINT statement, the 1st comma means the 1st tab zone, the 2nd comma the 2nd tab zone, the 3rd comma the 3rd tab zone, the 4th comma the 4th tab zone, and the 5th comma the 5th tab zone.

(3) Dollar sign (i.e. $)

The dollar sign does not mean money when it is used to form a string variable in BASIC. The sign tells the computer that it is a label to denote a string variable e.g. A$, Z7$, etc. There are 286 string variables used to be assigned string values (i.e. string constants). After the value-assignment, their values are stored in the computer memory. Dollar signs that are used to form string variables are usually found in LET, PRINT, INPUT, IF....THEN and READ statements. (Refer string variables for more information.)

(4) Equal sign (i.e. =)

The equal sign has its special meaning in LET statements which is different from its common usage in mathematics. Equal sign in the LET statement means 'replaced by'.

Examples

10 LET A=30

→ It means let 'A' be replaced by '30' (or assign 'A' to have the value of '30'.)

20 LET A=A+1

→ It means 'replace the value of A by the value of A+1'. Since the value of 'A' is 30 in line 10, therefore, the new value of 'A' in line 20 is 30 + 1 i.e. 31.

(Refer LET statement for more information.)

Related concepts

PRINT statement (p.95), LET statement (p.123), INPUT statement (p.135), DATA statement (p.160), String variable (p.66), string constant (p.48), tab zone (p.107)

PART III: BASIC STATEMENTS

SELF TEST

Note — If you can answer the following questions correctly, go to page 93. If not, go to next page.

Exercises

(1) What is the purpose of a REM statement?

(2) What is the difference when a REM statement is LISTed and RUN?

Answers

(1) The purpose of a REM statement is to allow the programmer to write a message to himself/herself or some one who may read the program some time in the future. The message usually serves to:-

(a) explain the program,

(b) describe the purpose of the program,

(c) define the meaning of some particular variables,

(d) clarify the upcoming part of a program, etc.

(2) When a REM statement is RUN, nothing comes out in the output because REM is not an executable statement. When a REM statement is LISTed, the same statement will be produced on the printout.

REM STATEMENT

Introduction

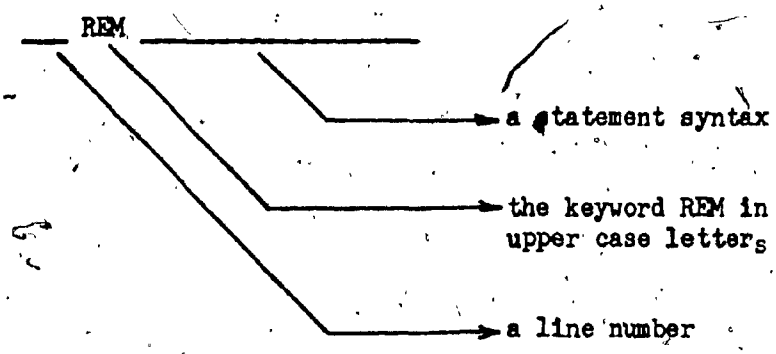
If you want to give a short introduction or explanation somewhere in your program, what should you do? Use the REM statement.

What is it?

REM statement is the short form of REMark. It is a comment statement. The purpose of REM statement is to allow the programmer to write a message to himself/herself or some one who may read the program some time in the future. The message usually serves to:-

(1) explain the program,
(2) describe the purpose of the program,
(3) define the meaning of some particular variables,
(4) clarify the upcoming part of a program, etc.

The format of REM statement is:-



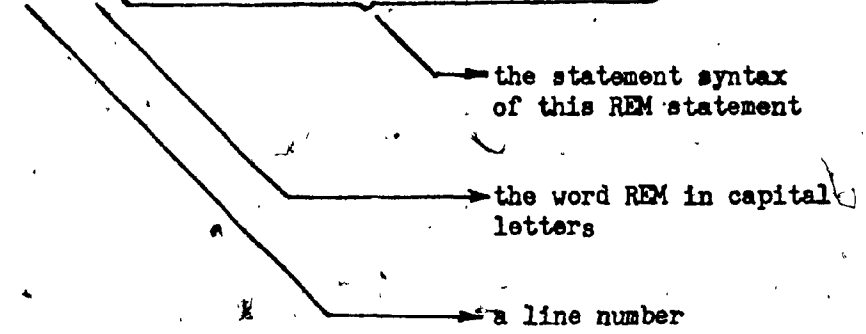
*Note: The two horizontal lines shown in the format are for illustrative purposes and they do not appear in your program.

Examples 1 .

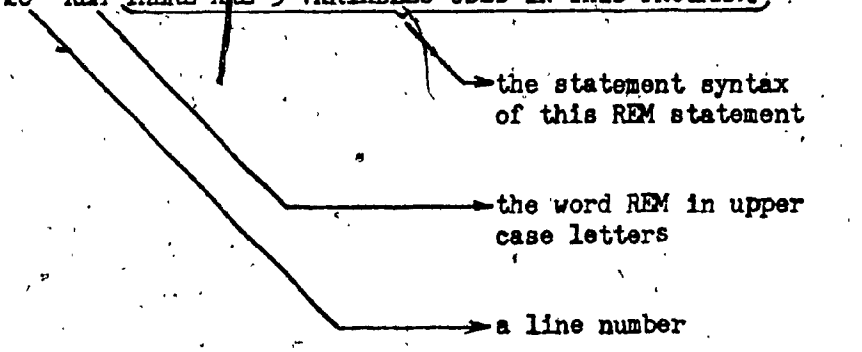
```
10  REM THIS IS A PROGRAM TO TEACH DIVISION.
20  REM THERE ARE 3 VARIABLES USED IN THIS PROGRAM.
30  REM  Y$="YES"
40  REM  N$="NO"
50  REM  M$="NAME"
```

Explanation

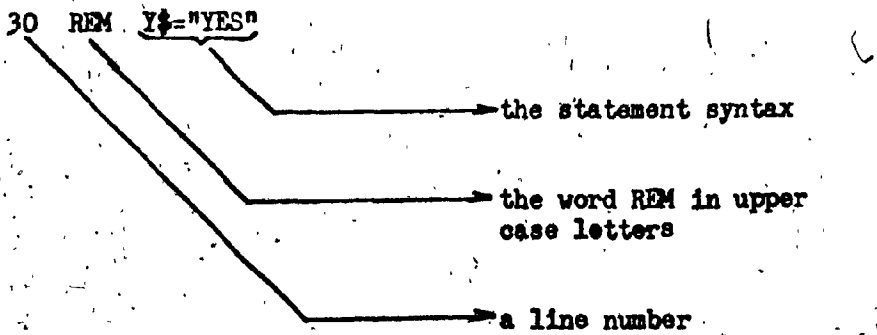
```
10  REM THIS IS A PROGRAM TO TEACH DIVISION.
```

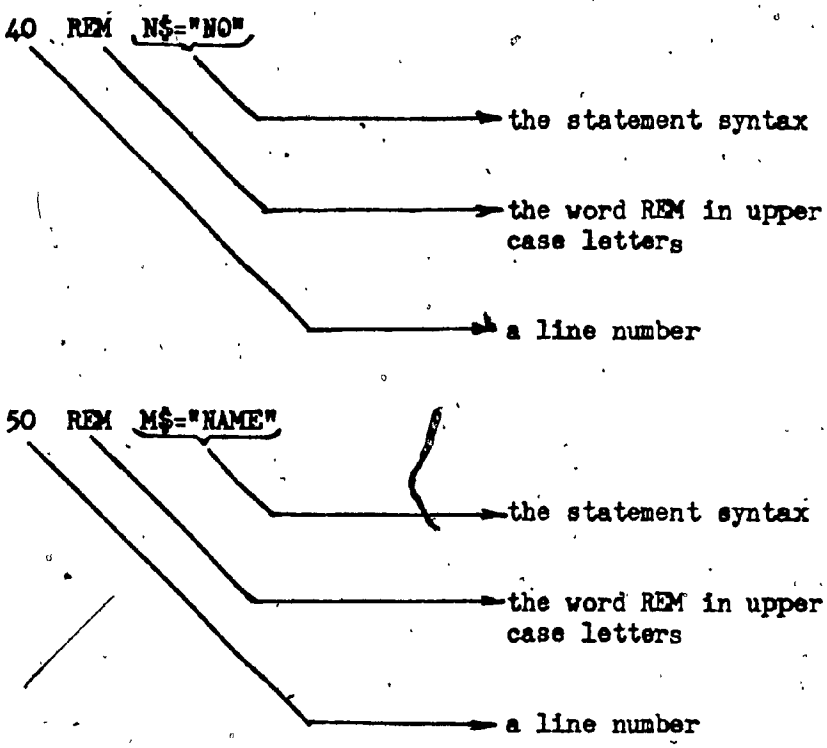



```
20  REM THERE ARE 3 VARIABLES USED IN THIS PROGRAM.
```




```
30  REM  Y$="YES"
```





All the above are REM statements. There is a line number before the word REM and a statement syntax after it. The statements serve as an explanation to clarify the upcoming part of the program. In programming, it is highly advisable to use as many REM statements as possible in your program.

Note

If you LIST the program, the computer would produce the same copy of REM statements you have typed in:-

```
LIST
10  REM THIS IS A PROGRAM TO TEACH DIVISION.
20  REM THERE ARE 3 VARIABLES USED IN THIS PROGRAM.
30  REM Y$="YES"
40  REM N$="NO"
50  REM M$="NAME"
```

But if you RUN the program, the computer does not print out any REM statements because the REM statements are non-executable statements which have no effect on the operation of the program. They only serve as an explanatory note in your program.

Exercises

(1) What is the purpose of REM statement?

(2) Type the following statements on the terminal. Then LIST and RUN the program.

```
10  REM THIS IS A PROGRAM TO COMPUTE
20  REM THE SQUARES OF SOME NUMBERS.
```

What is the difference when it is LISTed and RUN?

Answers

(1) REM statement is a REMark statement which is used to annotate or explain the upcoming part of a program. (or equivalent answer)

(2) When the program is LISTed, it appears as follows:-

```
10  REM THIS IS A PROGRAM TO COMPUTE
20  REM THE SQUARES OF SOME NUMBERS.
```

When the program is RUN, nothing comes out because REM is not an executable statement. That is to say it has no effect on the execution of the program.

SELF TEST

Note

This test determines how well you know the material in the coming map. If you can answer the following questions correctly, go to page 115. If not, go to next page.

Exercises

(1) Debug the following statements:-

```
PRINT My name is Henry.
PRANT I am 21 years old.
```

(2) Write out the printout result of the following statements:-

```
10 PRINT 5*10
20 PRINT
30 PRINT "5*10="; 5*10
40 PRINT "A";"B";"C"
50 PRINT "A","B","C"
```

(3) Code in BASIC by using the following information:-

(a) Print your name at the 1st tab zone, and your age at the 2nd tab zone.

(b) the answer 5 plus 2.

(c) the answer 10 divided by 2.

(d) the answer of 10 times 2.

Answers

(1) Revised statements

```
10 PRINT "My name is Henry."
20 PRINT "I am 21 years old."
```

(2) 50

```
5*10=50
ABC
A             B             C
```

(3) (a) 10 PRINT "C.R. BROWN", "Age: 25"

(b) 20 PRINT 5+2

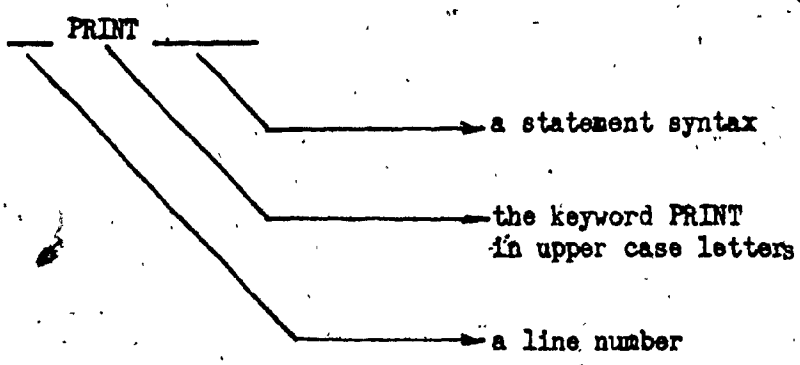
(c) 30 PRINT 10/2

(d) 40 PRINT 10*2

PRINT STATEMENT

What is it?

PRINT statement is a very common statement used in BASIC programming. It is used for inputting and outputting information. Its format is:-



The statement syntax may consist one or more of the followings:-

(1) a string constant,
(2) a numerical constant,
(3) a string variable,
(4) a numerical variable,
(5) an arithmetic expression,
(6) a blank.

*Note: The two horizontal lines in the format above are for illustrative purposes only and they do not appear in your program.

Functions of PRINT statement

There are two main functions of the PRINT statement. After we RUN the PRINT statement, the computer will:-

(A) produce or display the data information on the printout as mentioned in the statement syntax:-

(1) a string constant within the quotation marks,

(2) the value (i.e numerical constant) of a numerical variable(Note: The value is printed but not the variable.)

(3) the value (i.e. string constant) of a string variable (Note: The value is printed but not its variable.)

(4) the answer of an arithmetic expression,

(5) nothing if the syntax is a blank.

(B) determine the format of the output.

First function

(1) PRINTs out a string constant

If you want the computer to print out something, you have to type a line number, the word PRINT in upper case letter and the string constant inside the quotation marks. The quotation marks go in pairs and they serve to indicate to the computer where the string constant starts and ends. Then you have to type the word RUN and hit the RETURN key, and the computer will print out exactly the same text inside the quotation marks.

Examples 1

Input program

```
10  PRINT "HELLO!"
20  PRINT
30  PRINT "HOW ARE YOU?"
40  PRINT "500"
```

Hit the RETURN key after you have typed in every statement.

Explanation

```
10  PRINT "HELLO!"
```

- a string constant to be printed within the quotation marks
- the word PRINT in upper case letters
- a line number

```
20  PRINT
```

- a blank (no statement syntax)
- the word PRINT in upper case letters
- a line number

```
30  PRINT "HOW ARE YOU?"
```

- The whole question is a string constant including the question mark.
- the word PRINT in upper case letters

```
40 PRINT "500"
```

Note '500' here is not a numerical constant. It is a string constant because it is inside the quotation marks.

the word PRINT in upper case letters

a line number

Output program

```
RUN
HELLO!

HOW ARE YOU?
500
```

Explanation

RUN → Type RUN and hit the RETURN key.

HELLO! → The computer prints out the string constant.

→ Notice the computer prints nothing because of the blank in line 20.

HOW ARE YOU? → The computer prints out the question from line 30.

500 → It also prints out '500'.

Note

When you want the computer to print out a string constant you have to enclose it with opening and closing quotation marks. The computer does not tolerate any other format.

Non-examples 1

```
30 PRINT HELLO!
```

→ It is a syntax error because the word 'HELLO!' is not inside the quotation marks.

```
30 PRINT "HELLO!
```

→ It is a syntax error again because there is no closing quotation mark.

```
30 PRINT HELLO!"
```

→ It is a syntax error because there is no opening quotation mark.

```
30 PRINT 'HELLO!'
```

→ It is a syntax error because it it the wrong quotation marks. (*Note: some computer systems accept this kind of quotation marks.)

30 PRING "HELLO!"

It is a syntax error because the word 'PRINT' is mispelled.

PRINT "HELLO!"

It is a syntax error because there is no line number.

The above statements have bugs (syntax errors) and they have to be debugged before the computer can RUN the program for you.

Rule

To print a string constant, the PRINT statement must have a line number, the word PRINT in upper case letter and the string constant enclosed in both the opening and closing quotation marks.

First function (continued)

(2) PRINTs out a string constant mentioned in the LET statement

If you want the computer to print out a string constant mentioned in the LET statement, you have to type a line number, the word PRINT in upper case letter, and the string variable mentioned in the LET statement.

Example 2

Output program

```
10  LET H$="COME"
20  PRINT H$
```

Explanation

```
10  LET H$="COME"
```

→ a string constant in quotation marks.

→ a string variable

```
20  PRINT H$
```

→ the same string variable as mentioned in the LET statement

→ the word PRINT in upper case letters

→ a line number

Output program

```
RUN
COME
```

→ Notice the computer only prints out the string constant or the value of the string variable i.e. COME but not H$

Non-examples 2

```
10 LET 2$="COME"
20 PRINT 2$
```
→ It is a syntax error because the first character in the string variable cannot be a number.

```
10 LET $A="COME"
20 PRINT $A
```
→ It is a syntax error again because A must be followed by a dollar sign ($).

```
10 LET AC$="COME"
20 PRINT AC$
```
→ It is a syntax error because there are two alphbets before the dollar sign ($).

Rules

(1) A string variable has to be:-
(A) a letter plus a dollar sign, or
(B) a letter plus a digit plus a dollar sign.
The computer does not accept any other format.

(2) To print out a string constant from a LET statement, the PRINT statement must have a line number, the word PRINT in upper case letter, and the string variable mentioned in the LET statement.

First function (continued)

(3) PRINTs out a numerical constant mentioned in the LET statement

If you want the computer to print out the numerical constant mentioned in the LET statement, you have to type a line number, the word PRINT in upper case letter, and the numerical variable mentioned in the LET statement.

Example 3

Input program

```
10  LET A=50
20  PRINT A
```

Explanation

```
10  LET A=50
```

'50' is a numerical constant.

'A' is a numerical variable.

The above statement means A is assigned with the value of 50.

```
20  PRINT A
```

This 'A' is the same numerical variable in LET statement in line 10.

The above PRINT statement asks the computer to print out the value of A.

Output program

```
RUN
50
```

The value of the numerical variable 'A' is printed. (Note: 'A' is not printed out.)

Non-examples 3

```
30 LET 2A=10
40 PRINT 2A
```
→ It is a syntax error to use '2A' as a numerical variable because a letter must be followed by a digit.

```
10 LET AZ=10
20 PRINT AZ
```
→ It is a syntax error to use 'AZ' as a numerical variable because two letters are not allowed.

```
10 LET A26=10
20 PRINT A26
```
→ It is a syntax error to use 'A26' as a numerical variable because there are too many characters.

```
10 LET 33=10
20 PRINT 33
```
→ It is not accepted to use two digits like '33' as a numerical variable.

```
10 LET F#=10
20 PRINT F#
```
→ It is not allowable to use '#' or other punctation mark as numerical variable.

Rules

(1) To print a numerical constant from a LET statement, the PRINT statement must have a line number, the word PRINT in upper case letter, and the numerical variable.

(2) A numerical variable [illegible] to be:-

(A) a letter, or

(B) a letter plus a digit.

The computer does not accept any other format.

First function (continued)

(4) Prints out the answer of an arithmetic expression

Since BASIC has a built in computational operation, the PRINT statement can print out the result of arithmetic expressions. We can express an arithmetic expression by means of operators.

Examples 4

Input program

```
10 PRINT 20*20
20 PRINT 20+20
30 PRINT 20-20
```

Explanation

```
10 PRINT 20*20
```

→ This statement asks the computer to multiply 20 by 20 and print out the result.

```
20 PRINT 20+20
```

→ The computer is asked to add 20 to 20 and print out the result.

```
30 PRINT 20-20
```

→ The computer is asked to subtract 20 from 20 and print out the result.

Output program

```
RUN
 400
 40
 0
```

The computer executes all the arithmetic expressions and prints out the results.

Non-examples 4

```
10 PRINT 5**10
```

→ It is wrong to put two operators together.

```
10 LET A=50
20 LET B=7
30 PRINT AB
```

→ It is not correct to express multiplication as 'AB'. The correct way is 'A*B'.

Rules

(1) To print the answer of arithmetic expression, the PRINT statement must have a line number, the word PRINT in upper case letters, and the arithmetic expression.

(2) In presenting arithmetic expression, bear in mind the priority of computation done by the computer.

Second function

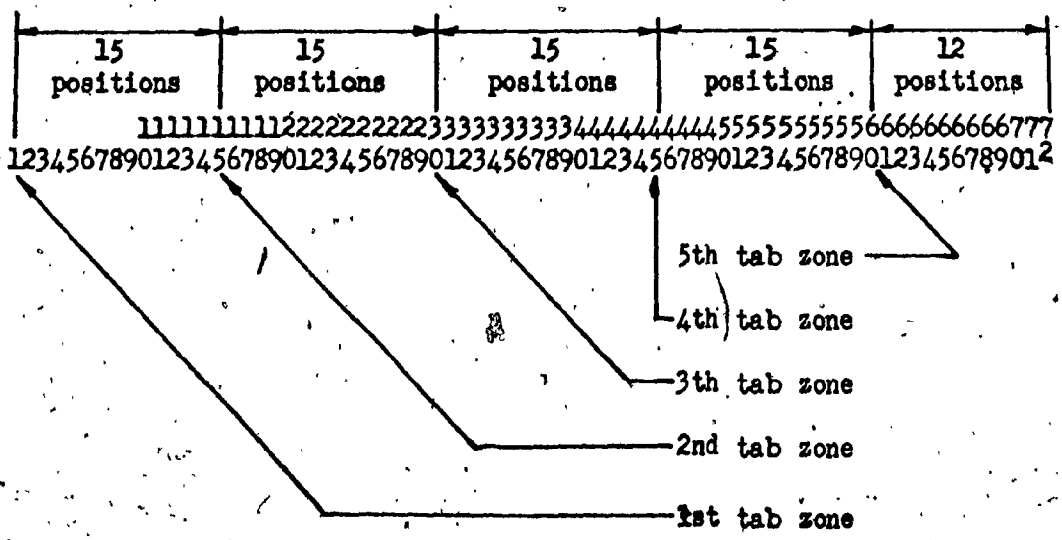
(B) Determines the printing format

Besides printing data output of the above nature, the PRINT statement also can tabulate the printing position of your output program by using the following 2 punctuation marks:-

(1) Comma (,), which means there are 5 tab zones to move into in one line.

(2) Semicolon(;), which means squeezing together in the print format (i.e. no printing space).

In order to understand the tabulating features of a computer terminal, you have to go there and try it out. Most of the computer systems have 72 printing spaces (character positions or character spaces). That is to say you can type 72 characters on the terminal in one line. The computer divides the 72 spaces into 5 zones, the first 4 zones with 15 spaces and the last zone with 12 spaces. If you want to test it on the computer terminal, you can try something like this:-



When you type a comma (,) after an item in your PRINT statement, the computer prints out that item at the first tab zone. If you type a comma after the second item, the computer prints out that item at the second tab zone, etc. But don't type any punctation mark after the last item (see e.g. 8 on p.92)

Example 5 Input program

```
10 PRINT "A","B","C","D","E"
```

This is the 5th item in this PRINT statement. Don't put any punctation after this item.

This is the 4th item and there is a comma after it. The comma tells the computer to print out this item at the 4th tab zone.

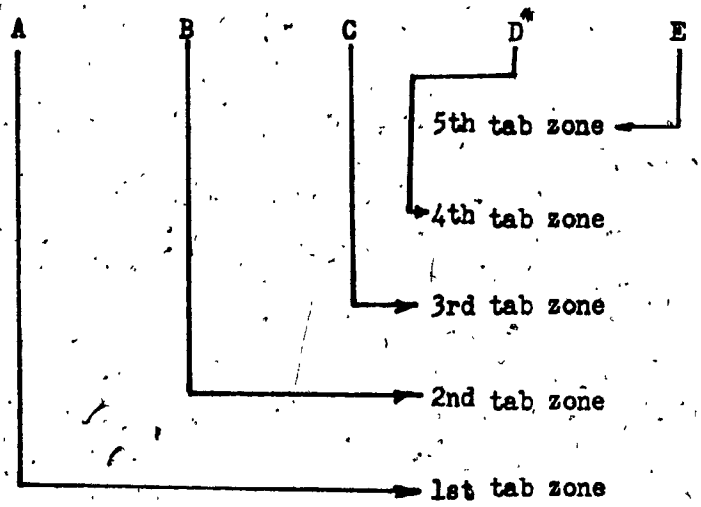
This is the 3rd item and there is a comma after it. It tells the computer to print out this item at the 3rd tab zone.

This is the 2nd item and there is a comma after it and it tells the computer to print out this item at the 2nd tab zone.

This is the 1st item and there is a comma after it. The comma asks the computer to print out this item at the first tab zone.

Output program

```
RUN
A          B          C          D          E
```



Example 6

Input program

```
10  PRINT "A";"B";"C";"D";"E"
```

Notice there is a semicolon after each item except the last item. The semicolons ask the computer to squeeze all the items together. There is no punctation mark after the last item.

Output program

```
RUN
ABCDE
```

All the items are squeezed together.

*Note: (1) The text inside the brackets after the word 'PRINT TAB' can be a numerical variable or an arithmetic expression. In this case the computer works out the result for the PRINTing position.

(2) The two horizontal lines shown in the format are for illustrative purposes and they do not appear in your program.

Example 1

Input program

```
10  PRINT TAB(7);"A"
```

Explanation

```
10  PRINT TAB(7);"A"
```

- "A" → a string constant 'A' inside the quotation marks
- ; → It is a must to type a semi-colon after the brackets.
- (7) → The number '7' inside the brackets asks the computer to print the string constant 'A' at the 7th printing position.
- PRINT TAB → the words 'PRINT TAB' in upper case letters
- 10 → a line number.

Output program

```
RUN
       A
```

'A' is printed at the 7th printing position.

Example 2

PRINT TAB is a very useful statement because you can ask the computer to print your items anywhere you like.

Input program

```
50  PRINT TAB (10);"A" TAB (25);"B" TAB (40);"C"
```

Explanation

```
50  PRINT TAB (10);"A" TAB (25);"B" TAB (40);"C"
```

This asks the computer to print 'C' at the PRINT position 40.

This asks the computer to print 'B' at the PRINTing position 25.

This asks the computer to print 'A' at the PRINTing position 10.

Output program

```
RUN
A          B          C
```

Rules

(1) Number your PRINT TAB in the brackets for PRINTing position from 0 to 71 (*Note: Some other computer systems may not have 72 print positions). Never use any PRINTing position that is greater than 71

because there are only 72 printing positions in one line.

(2) The number in PRINT TAB brackets must be presented in an increasing sequence, or the computer will give you erroneous output.

(3) Some computer systems use the nearest integer if you use an arithmetic expression inside the PRINT TAB brackets e.g. TAB (5*1.25) as TAB (7), whereas many systems take the integral part.

Non-example 1

Input program

```
10  PRINT TAB (30);"A" TAB (20);"B" TAB (10);"C"
```

Output program

```
RUN
                              A
                              PRINTing position 30
```

Explanation

Notice the computer only prints 'A' at the 30th PRINTing position. It does not print 'B' and 'C' at PRINTing positions 20 and 10. Why? It is because the computer does not move the teletypewriter back to the left. Since it has tabulated PRINTing position 30 it does not tabulate back to the positions 20 and 10. Therefore, you have to present your numbers inside the PRINT TAB brackets in increasing sequence.

Non-examples 2

50 PRINT TAB(50)"GOOD" ——→ It is a syntax error not to have a semicolon after '(50)'.

80 PRINT 50 TAB "W" ——→ It is wrong to put '50' in front of 'TAB' and not in parentheses.

Summary

PRINT TAB statement is used to determine the exact printing position of your output program.

Exercises

(1) Use just one PRINT TAB statement to print out the following data:-

(a) Print A at the 5th printing position.

(b) Print B at the 10th printing position.

(c) Print C at the 25th printing position.

(d) Print D at the 30th printing position.

*Note: 'A', 'B', 'C' & 'D' are string constants.

(2) Code the follgwing by using LET & PRINT TAB statements (Study LET statement before you answer this question):-

A's value is 30.

B's value is YES.

C's value is NO.

(a) Print A's value at the 5th printing position.

(b) Print B's value at the 10th printing position.

(c) Print C's value at the 25th printing position.

*Note: 'A' is a numerical variable while 'B' & 'C' are string variables.

Answers

(1)

Input program

```
10 PRINT TAB(5);"A" TAB(10);"B" TAB(25);"C" TAB(30);"D"
```

Output program

```
RUN
    A     B                C     D
```

(2)

Input program

```
10 LET A=30
20 LET B$="YES"
30 LET C$="NO"
40 PRINT TAB (5);A TAB(10);B TAB(25);C
```

Output program

```
RUN
     30    YES              NO
```

Related concepts

PRINT statement (p.95), LET statement (p.123)

SELF TEST

Note

If you can answer the following questions correctly, to to page 134. If not, go to next page.

Exercises

(1) Code the following in BASIC:-

(a) Let A have the value '20'.

(b) Let B have the value '30'.

(c) Let A$ have the value 'pens'.

(d) Let B$ have the value 'ball pens'.

(e) Print out the information as follows:-
20 pens (at the 1st tab zone), & 30 ball pens (at the 2nd tab zone).

(2) Debug the following statements:-

```
10 A=5
20 B="GOOD"
30 C$=5
40 $C="BAD"
```

Answers

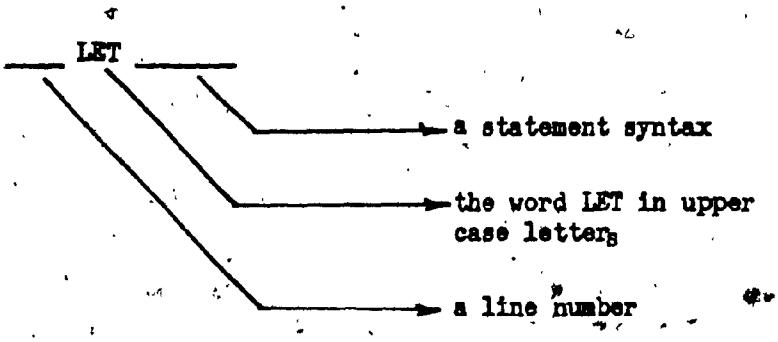
(1)
```
10 LET A=20
20 LET B=30
30 LET A$="PENS"
40 LET B$="BALL PENS"
50 PRINT A;A$, B;B$
```

(2) line 10: no bug
```
20 B$="GOOD"
30 C=5
40 C$="BAD"
```

LET STATEMENT

What is it?

Assigning a value to a variable is a common practice in BASIC programming. LET is called an assignment statement. It is used to assign values to either numerical or string variables. Its format is:-



*Note: The two horizontal lines in the format above are for illustrative purposes and do not appear in your program.

Rule

In assigning numerical values to numerical variables, you can use 286 numerical variables in your program. They are formed by:-

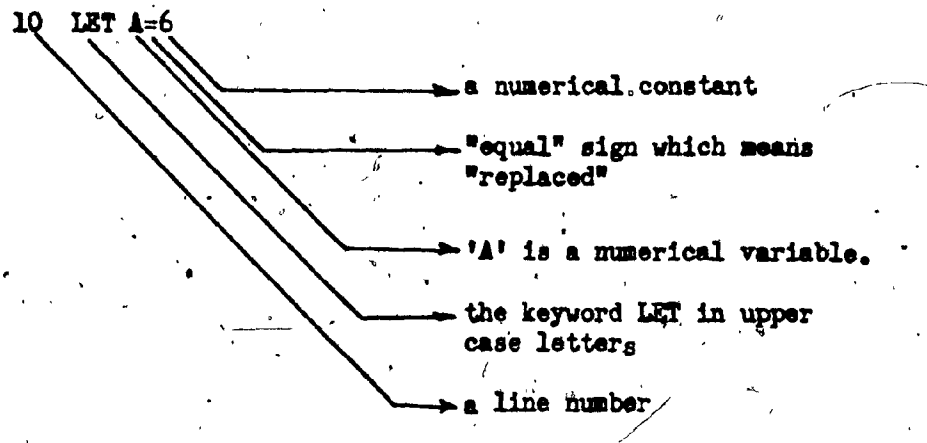
(A) any letter, or
(B) any letter plus one digit only.

Example 1

Input program

```
10 LET A=6
20 PRINT A
```

Explanation



The above statement means: let 'A' have the value '6' and store in the computer memory.

20 PRINT A → This statement asks the computer to print the value of 'A'. (Note: The computer is not going to print 'A'.)

Output program

```
RUN
6
```

→ The value of the numerical variable 'A' is printed.

Example 2

Input program

```
10 LET A=5
20 LET B=10
30  LET A=A+1
40  PRINT A,B,A*B
```

Explanation

Statement	Numerical variable & value		Explanation
10 LET A=5	A	5	'A' has the value 5 but 'B' has 0.
	B	0	
20 LET B=10	A	5	The value of 'A' is still 5 but the value of 'B' is 10
	B	10	
30 LET A=A+1	A	5+1	The value of 'A' is changed by adding 1 to 5 i.e. 6. The value of 'B' is still 10.
	B	10	
40 PRINT A,B,A*B	A	6	'A' is 6 and 'B' is 10 (same as in line 30).
	B	10	

Output program

RUN

6 10 60

60 → the answer of 6 x 10 at 3rd PRINT position.

10 → the value of the numerical variable 'B'.

6 → the value of the numerical variable 'A'

Special notes:

Study line 40 carefully:

```
40  PRINT A,B,A*B
```

There are several special points in this statement:-

(1) It asks the computer to print the values of 'A' & 'B' and the answer of 'A' times 'B', but not the variables.

(2) 'A', 'B' and 'A*B' are numerical variables.

(3) The value of 'A' in line 40 is 6 which has replaced the initial value of 'A' (i.e. 5) at line 10. A variable is only allowed to have one value at a time, therefore, the old value is discarded in the memory cell of the computer while the new value is in use.

(4) The use of a comma after 'A' and 'B' in line 40 determines the PRINT position of its output.

Example 3

In BASIC, if the variable is not assigned a value, its value is considered as zero.

Input program

```
10  LET A=10
20  LET B7=20
30  LET C=A+B7+D
40  PRINT A, B7, C,D
```

Explanation

Statement	Numerical variable & value	Explananion
10 LET A=10	A 10	The value of 'A' at line 10 is 10.
20 LET B7=20	A 10 B7 20	The value of 'A' is still 10 and the value of 'B7' is 20.
30 LET C=A+B7+D	A 10 B7 20 C 30 D 0	The value of 'A' is 10, 'B7' is 20, 'C' is 30 (i.e. 10+20+0) and 'D' is zero.

Output program

```
RUN
10          20          30          0
                                    ↓
                                    D's value
                        C's value ←
            B7's value ←
A's value ←
```

Note: The value of 'D' in line 30 is not assigned and it is given as zero.

Example 4

In some BASIC systems the word LET is often omitted and the computer still performs the same job for you.

Input program

```
10  A=5
20  B=10
30  C=2
40  PRINT A,B,C,A*B*C
```

Output program

```
RUN
5             10            2             100
```

Rule

In assigning string values to string variables, you can use 286 string variables in your program. They are formed by:-

(A) a letter plus a dollar sign, or

(B) a letter plus a digit plus a dollar sign.

Example5

Input program

```
10  LET H$="HELLO!"
20  LET I$="IT'S NICE TO MEET YOU."
30  LET A$="HOW ARE YOU?"
40  PRINT H$
50  PRINT I$
60  PRINT A$
```

Explanation

10 LET H$="HELLO!" → The string value of the string variable 'H$' is 'HELLO!'. Note that the the string constant is enclosed in quotation marks.

20 LET I$="IT'S NICE TO MEET YOU." → The string value of the string variable 'I$' is 'IT'S NICE TO MEET YOU.'

30 LET A$="HOW ARE YOU?" → The string value of the string variable 'A$' is 'HOW ARE YOU?'.

40 PRINT H$ → This asks the computer to print out the value of 'H$'.

50 PRINT I$ → The computer is asked to print out the value of 'I$'.

60 PRINT A$ → The computer is asked to print out the value of 'A$'.

*Note: All the string constants in line 10, 20 & 30 are enclosed in quotation marks.

Output program

```
RUN
HELLO!
IT'S NICE TO MEET YOU.
HOW ARE YOU?
```

Rules

(1) To assign a string value, the LET statement must have a line number, the word LET in upper case letter, a string variable, an "equal" sign (=), and a string constant in quotation marks.

(2) To assign a numerical value, the LET statement must have a line number, the word LET in upper case letter, a numerical variable, an "equal" sign (=), and a numerical constant.

(3) You are <u>not</u> allowed to assign a string value to a numerical variable, <u>or</u> a numerical value to to a string variable.

Non-example 1

10 LET A$=5 ——→ It is a syntax error because 'A$' is a string variable and '5' is a value of a numerical variable (i.e. a numerical constant).

You can debug it as follow:-

10 LET A$="5" ——→ In this case '5' is a string constant.

<u>or</u>,

10 LET A=5 ——→ In this case 'A' is a numerical variable.

Non-example 2

10 LET B="HOUSE" ——→ It is a syntax error because 'B' is a numerical

variable and 'HOUSE' is a string value.

You can debug it as follow:-

10 LET B$="HOUSE" ← In this case 'B$' is a string variable and 'HOUSE' is a string constant.

Summary

(1) LET is an assignment statement. It is used to assign a value to a variable.

(2) The "equal" sign in a LET statement does not mean an algebraic equation. It means replacing the value.

(3) Besides assigning values, the LET statement can also be used to calculate an arithmetic expression.

(4) A string value should be assigned to a string variable and a numerical value should be assigned to a numerical variable.

Exercises

(1) Find all the bugs in the following statements and debug them.

```
410 A=5
420 LET BC=10
420 LET F=FALL
430 LAT G$="GOOD
```

(2) Design a program by using the following data:-

A has the value of 10.

B has the value of 20.
C has the value of 50.
D is equal to A+B+C.

Print all the values A,B,C & D on one line with 10 printing position separated from each value.
(Note: 'A', 'B', 'C' & 'D' are numerical variables.)

Answers

(1) 1st line: no bug (see p.106 for explanation)
2nd line: 'BC' is incorrectly used as a numerical variable.
3rd line: The line number is exactly the same as the last one. It should be 430. 'F' is a numerical variable but 'FALL' is a string constant.
4th line: The word 'LET' is mispelled and a closing quotation mark is missing for the string constant.

Revised program

```
410 A=5
420  LET B=10
430  LET F$="FALL"
440  LET G$="GOOD"
```

(2) Input program

```
10  LET A=10
20  LET B=20
30  LET C=50
```

```
40  LET D=A+B+C
50  PRINT TAB(0);A TAB(10);B TAB(20);C TAB(30);D
```

Output program

```
RUN
10          20          50          80
```

Related concepts

Numerical constant (p.8), string constant (p.48),

Numerical variable (p.60), String variable (p.66)

SELF TEST

Note — If you can answer the following questions correctly, go to page 159. If not, go to next page.

Exercises

(1) What is the difference between LET & INPUT statements?

(2) Explain why a PRINT statement(s) is (are) necessary before a INPUT statement.

(3) Code in BASIC by asking a user's name and age. (Store them in the computer memory.)

(4) Write a program to calculate the square of a number. (The user is going to supply the number while the computer is going to type out the square of that number.)

Answers — See page 156.

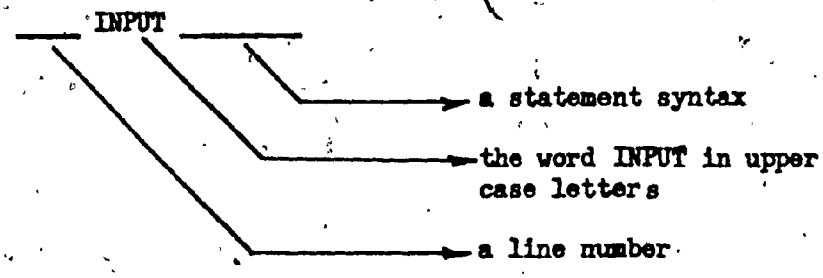
INPUT STATEMENT

Introduction

Both LET and INPUT statements are used to assign values to variables. The LET statement is used by the programmer to assign values to variables when he/she is designing the program. But the INPUT statement is used when the user is running the program.

What is it?

The INPUT statement is a BASIC statement which is designed by a programmer where a user assigns a value to a variable at the time the user runs the program in the computer. Note carefully that the value is <u>not</u> coded (assigned) into the program when it is designed. Instead, the value is given at the time when the program is RUN. The INPUT statement can be used to assign a value to either numerical or string variables by the user. Its format is:-



*Note:- (1) A statement syntax can be one of the followings:-

(A) one or more numerical variables,

(B) one or more string variables, or

(C) one or more string and numerical variables.

(2) The two horizontal lines shown in the format are for illustrative purposes and they do not appear in your program.

Example 1 (1) Assigning a value to one or more numerical variables

Input program

```
10 PRINT "PLEASE TYPE AN ARABIC NUMBER."
20 INPUT N
30 PRINT "THE NUMBER YOU HAVE JUST TYPED IS";N
```

Explanation of the input program

10 PRINT "PLEASE TYPE AN ARABIC NUMBER."

→ The computer is asked to print out the string constant in quotation marks.

20 INPUT N

→ 'N' is a numerical variable.

→ the word 'INPUT' in upper case letters

→ a line number

Note: When this statement is executed the user will be asked to type a number on the terminal and the value of this number is assigned as 'N'.

30 PRINT "THE NUMBER YOU HAVE JUST TYPED IS";N

→ 'N' is the same numerical variable in line 20.

Line 30 asks the computer to print out the string constant in the quotation marks and also the value of 'N'. Still remember the use of semicolon (;) after the closing quotation mark? It means squeezing together the printing space.

Output program

```
RUN
PLEASE TYPE AN ARABIC NUMBER.
?7
THE NUMBER YOU HAVE JUST TYPED IS 7
```

Explanation of the output program

```
RUN
PLEASE TYPE AN ARABIC NUMBER.
```

The user is asked to type an Arabic number.

```
?7
```

The user types '7' and RETURN key.

When line 20 is executed the computer prints out a question mark (?) asking the user to type a number. The computer sits and waits for you to type in an answer. If you don't answer the computer will wait and shut down after a while.

```
THE NUMBER YOU HAVE JUST TYPED IS 7
```

The computer prints out the string constant and the number you have typed in.

*Note:
'N' in the INPUT statement of line 20 is a numerical variable that the programmer designed so that the user of this program can give an input value to it. After the user has typed in the number '7' (it has to be in Arabic form, if it is typed as 'seven' then it is a string value) the value of 'N' is assigned as '7'. Inside the computer the memory cell is stored as below:-

Numerical variable	Value
N	7

Example 2

You can use the INPUT statement to ask the user to give more than one numerical value.

Input program

```
10 PRINT "TYPE 3 ARABIC NUMBERS.  USE"
20 PRINT "COMMA TO SEPARATE EACH NUMBER."
30 INPUT A,B,C
40 PRINT "THE NUMBERS YOU HAVE TYPED ARE";A;B;C
```

Explanation

```
10 PRINT "TYPE 3  ARABIC NUMBERS.  USE"
20 PRINT "COMMA TO SEPARATE EACH NUMBER."
```

These two statements ask the computer to print out the message inside the quotation marks.

```
30 INPUT A,B,C
```

The programmer uses 3 numerical variables: 'A', 'B' & 'C'. You have to use commas to separate each variable.

```
40  PRINT "THE NUMBERS YOU HAVE TYPED ARE";A;B;C
```

The computer is asked to print out the string constant and the values of 'A', 'B' and 'C'.

Output program

```
RUN
TYPE 3 ARABIC NUMBERS.  USE
COMMA TO SEPARATE EACH NUMBER.
?9,8,1
THE NUMBERS YOU HAVE TYPED ARE 981
```

Explanation of the output program

```
RUN
TYPE 3 ARABIC NUMBERS.  USE
COMMA TO SEPARATE EACH NUMBER.
```

The user is asked to type 3 numbers.

```
?9,8,1
```

These are the 3 numbers you typed in.

The computer is waiting for your answer. It will shut off if you don't answer.

```
THE NEMBERS YOU HAVE TYPED ARE 9 8 10
```

The computer prints out the string constant and the values of the 3 numerical variables.

*Note:-
One single INPUT statement can request a user to type in several numerical values. In the above example the assignment of values to the numerical variables is as follows:-

Numerical variable	Value
A	9
B	8
C	10

Actually it means:

```
LET A=9
LET B=8
LET C=10
```

But all these values are assigned by the user of the program.

Example 3

(2) Assigning value to one or more string variables

Input program

```
10 PRINT "TYPE A WORD."
20 INPUT A$
30 PRINT A$;" IS THE WORD YOU HAVE TYPED."
```

Explanation

```
10 PRINT "TYPE A WORD."
```

This asks the computer to print out the string constant inside the quotation marks.

```
20 INPUT A$
```

The programmer uses 'A$' as a string variable to be assigned with a value when the user types in a text.

```
30 PRINT A$; " IS THE WORD YOU TYPED."
```

This asks the computer to print out the value of the string variable and the string constant in the quotation marks.

Note: There is a space after the opening quotation mark.

Output program

```
RUN
TYPE A WORD.
?BAD
BAD IS THE WORD YOU TYPED.
```

Explanation

RUN

TYPE A WORD. → The user is asked to type a word on the terminal.

?BAD

→ You type 'BAD' and hit the RETURN key.

→ The computer prints out the question mark (?) and waits for your answer.

BAD IS THE WORD YOU TYPED. → The computer prints out both the value of the string variable and the string constant.

*Note:

A string variable always carries a dollar sign at the end just like A$. Only string values can be assigned to string variables. After the assignment is done, the string value is stored in the computer as follows:-

String variable	Value
A$	BAD

It means:

LET A$="BAD"

Remember: the value is assigned by the user of the program.

Example 4 A single INPUT statement can also request a user to give several string values.

Input program

```
10 PRINT "TYPE 3 ADJECTIVES AND USE A COMMA"
20 PRINT "TO SEPARATE EACH ADJECTIVE."
30 INPUT A$,B$,C$;
40 PRINT A$;B$;C$;" ARE THE 3 ADJECTIVES YOU TYPED."
```

Explanation of the input program

```
10 PRINT "TYPE 3 ADJECTIVES AND USE A COMMA"
20 PRINT "TO SEPARATE EACH ADJECTIVE."
```

The computer is asked to print out the string constant in lines 10 & 20.

```
30 INPUT A$,B$,C$
```

The programmer user 'A$', 'B$' & 'C$' as string variables to be assigned string values.

```
40 PRINT A$;B$;C$;" ARE THE 3 ADJECTIVES YOU TYPED."
```

The computer is instructed to print out the values of the 3 string values (after the values are typed in by the user) and the string constant in the quotation marks.

Output program

```
RUN
TYPE 3 ADJECTIVES AND USE A COMMA
TO SEPARATE EACH ADJECTIVE.
?BAD, GOOD, HOT
BAD GOOD HOT ARE THE 3 ADJECTIVES YOU TYPED.
```

Explanation of the output program

```
RUN
TYPE 3 ADJECTIVES AND USE A COMMA
TO SEPARATE EACH ADJECTIVE.
```

The computer prints out the message asking the user to type in 3 adjectives.

```
?BAD,GOOD,HOT
```

You type the 3 adjectives and hit the RETURN key.

The computer prints out the question mark (?) asking the user to give the answers. If you don't answer, the computer will wait for a while and shut off.

```
BAD GOOD HOT ARE THE 3 ADJECTIVES YOU TYPED.
```

The computer prints out the values of the 3 string values and the string in the quotation marks.

*Note:
The INPUT statement (line number 30) is designed to ask the user of the program to provide 3 string values. After the 3 values have been typed in, their values are assigned and stored in the memory of the computer as follow:-

String Variable	Value
A$	BAD
B$	GOOD
C$	HOT

That is to say:-

```
LET A$="BAD"
LET B$="GOOD"
LET C$="HOT"
```

All the above values are assigned by the user. The function of INPUT statement is exactly the same as LET statement except that the values are only assigned by the user during the execution of the program. However, the assignment of values in LET statement is done by the programmer when he is designing the program.

Example 5

The last line in the output program of the above example is:-
'BAD GOOD HOT ARE THE 3 ADJECTIVES YOU TYPED.'
If you want to put comma after the word 'BAD' and 'GOOD' and put the word 'AND' before 'HOT', what should you do? This can be done as follow:-

Input program

```
40 PRINT A$;", ";B$;", AND";C$;" ARE THE 3 ADJECTIVES YOU TYPED.
```

Output program

```
TYPE 3 ADJECTIVES AND USE A COMMA
TO SEPARATE EACH ADJECTIVE.
?BAD, GOOD, HOT
BAD, GOOD, AND HOT ARE THE 3 ADJECTIVES YOU TYPED.
```

Explanation

```
40 PRINT A$;", ";B$;", AND";C$;" ARE THE 3 ADJECTIVES YOU TYPED.
```

Put a space here.
Put a space here.
Put a comma here.
Put a space here.
Put a comma here.

*Note: Quotation marks are very useful in arranging a better format for your output. Also note the use of semicolons in the above statement.

Example 6

(3) Assigning to both numerical and string values in the same line

INPUT statements can be designed by a programmer to call for both numerical and string values from a user when the program is executing.

Input program

```
10 PRINT "PLEASE TYPE YOUR NAME FIRST THEN"
20 PRINT "YOUR AGE IN ARABIC NUMBER.  SEPARATE YOUR"
30 PRINT "NAME AND AGE BY A COMMA."
40 INPUT N$,A
50 PRINT N$; ", YOU'RE ";A; " YEARS OLD.  IT'S NICE"
60 PRINT "TO MEET YOU."
```

Explanation

```
10 PRINT "PLEASE TYPE YOUR NAME FIRST THEN"
20 PRINT "YOUR AGE IN ARABIC NUMBER.  SEPARATE YOUR"
30 PRINT "NAME AND AGE BY A COMMA."
```

These 3 statements ask the computer to print out the string constant in quotation marks.

```
40 INPUT N$,A
```

The programmer uses 'N$' as a string variable for the user's name and 'A' as a numerical variable for his/her age.

```
50 PRINT N$; ", YOU'RE ";A; " YEARS OLD.  IT'S NICE"
60 PRINT "TO MEET YOU."
```

The computer is asked to type out the values of 'N$' and 'A' after they are filled in by the user, and also asked to print out the string constant

within the quotation marks. Note the use of a semicolon after the numerical variable 'N$' and string variable 'A'. Note the use of a space inside the quotation marks.

Output program

```
PLEASE TYPE YOUR NAME FIRST THEN
YOUR AGE IN ARABIC NUMBER.  SEPARATE YOUR
NAME AND AGE BY A COMMA.
?PIERRE,20
PIERRE, YOU'RE 20 YEARS OLD.  IT'S NICE
TO MEET YOU.
```

Explanation of the output program

```
RUN
PLEASE TYPE YOUR NAME FIRST THEN
YOUR AGE IN ARABIC NUMBER.  SEPARATE YOUR
NAME AND AGE BY A COMMA.
```

The message asks the user to type his/her name and age.

```
?PIERRE,20
```

The user typed in the answers as requested.

The computer prints out a question mark (?) asking for the user's response.

Remember: The computer will shut off if you don't hit the RETURN key after you have typed in because the computer does not know whether you have finished or not.

```
PIERRE, YOU'RE 20 YEARS OLD.  IT'S NICE
TO MEET YOU.
```

See how beautifully this line is done by the line 50 & 60! See the use of semicolon, space and quotation marks in these 2 statements

*Note:

```
40 INPUT N$,A
```

In this statement, 'N$' is a string variable and 'A' is a numerical variable. 'N$' is typed in first, and 'A' right after it. Notice the user also typed in 'his name' first and 'his age' right after:-

```
?PIERRE,20
```

'His age' typed in later.

'His name' typed in first.

Which is typed in first or later is very important to the computer. If the user does not obey the instruction as told (e.g. he types in his age first and name late) there will be a syntax error. If the user types in the information as above the computer would register its

memory as follows:-

	Variable	Value
String	N$	PIERRE
Numerical	A	20

That is to say the assignment of values is:-

LET N$="PIERRE"

LET A=20

Note

In every preceding example the INPUT statement is always preceded with one or more PRINT statements. The PRINT statement(s) serve(s) an an explanatory message or a command asking what the user should do or should type. This sets a limit for what the user will be typing in. Therefore, the message (the string constant) in the PRINT statement(s) has to be carefully designed and the user has to pay great attention to the instruction. If not, it would render a syntax error. If the user types in something wrong (which is against the syntax of BASIC) the computer will not execute the program until the proper instruction is typed in.

Non-example 1

Kind of bug: not enough data

Input program

```
10 PRINT "TYPE 3 ARABIC NUMBERS.  USE A COMMA TO"/
20 PRINT "SEPARATE EACH OF THEM."
30 INPUT A,B,C
```

Output program

```
RUN
TYPE 3 ARABIC NUMBERS.  USE A COMMA TO
SEPARATE EACH OF THEM.
?5,7
NOT ENOUGH DATA; RETYPE LINE
```

Explanation of the output program

As you can see the input fed in by the user:-

?5,7 → The user just typed in 2 numbers.

NOT ENOUGH DATA; RETYPE LINE → The computer refuses to execute because there are not enough data and the user has to enter one more number.

Non-example 2

Kind of bug: too much data

With the same program above, if the user enters too much data, the computer will refuse to execute.

Output program

```
RUN
TYPE 3 ARABIC NUMBERS.  USE A COMMA TO
SEPARATE EACH OF THEM.
?2,3,4,5,6,7
```

```
TOO MUCH DATA;  RETYPE LINE
```

Explanation of the output program

```
?2,3,4,5,6,7 ————————————→ The user types in more than
                            3 numbers.

TOO MUCH DATA;  RETYPE LINE → The computer gives out the
                              error message.  The user has
                              to retype the line again.
                              (*Note:  Some systems accept
                              the 3 typed in first numbers
                              and executes the program and
                              ignores the rest of the numbers.)
```

Non-example 3

Kind of bug: Mismatching

Mismatching means the typed in value does not match the variable designed in the INPUT statement.

Input program

```
10 PRINT "PLEASE TYPE 2 ADJECTIVES FIRST."
20 PRINT "THEN TYPE 2 ARABIC NUMBERS."
30 PRINT "SEPARATE EACH ITEM BY A COMMA."
40 INPUT A$,B$,C,D
```

Output program

```
RUN
PLEASE TYPE 2 ADJECTIVES FIRST.
THEN TYPE 2 ARABIC NUMBERS.
```

```
?5,20,GOOD,BAD
WRONG KIND OF DATA;  RETYPE LINE
```

Explanation of the output program

```
?5,20, GOOD, BAD ──────────→ The user types in 2
                             numbers first, then 2
                             adjectives.

WRONG KIND OF DATA;  RETYPE LINE
                    ↘
                      The data are entered in the
                      wrong sequence.  The user
                      has to type 2 adjectives
                      first, then 2 numbers.
```

*Note:

The reason they are the wrong kind of data is that the designer of the program has already designed the first 2 items as string variables and the others as numerical variables in line 40:-

```
40 INPUT A$,B$,C,D
          ↘  ↘
            'C' & 'D' are numerical
            variables and they need
            numerical values (constants)
            to match them.

            'A$' & 'B$' are string variables
            and they need string values
            (constants) to match them.
```

Therefore, special attention should be paid when assigning

values to string and numerical variables. Only a numerical value can be assigned to a numerical variable and only a string value can be assigned to a string variable.

Rule

From the above non-examples, you can see the importance of PRINT statement(s) used before the INPUT statement. When you design a program you should use clear and precise instructions in your PRINT statement(s) so that the user does not misinterpret your message. Thus, to avoid ambiguities, try not to use imprecise strings in the PRINT statement such as:-

(1) 10 PRINT "PLEASE TYPE A NUMBER."
The word 'number' is misleading because the user may type 'twenty' instead of '20'. 'Twenty' is a string value but '20' is a numerical value.

(2) 10 PRINT "TYPE 3 NUMBERS."
In this statement you should ask the user to use a comma to separate the numbers because the computer might misunderstand how many numbers the user has typed in.

(3) 10 PRINT "TYPE YOUR NAME AND YOUR AGE."
In response to this statement, the user may type:-
'JOHN BROWN, 20'.....(i) or,
'MY NAME IS JOHN BROWN AND I AM 20 YEARS OLD.'.....(ii)
The two responses are different. In (i) 'JOHN BROWN' is a string value and '20' is a numerical

value. But in (ii) the whole sentence is a string value.

Rules

(1) To assign a string value(s) to a string variable(s) by using the INPUT statement, the INPUT statement must have a line number, the word 'INPUT' in upper case letters and the string variable(s).

(2) To assign a numerical value(s) to a numerical variable(s) by using the INPUT statement, the INPUT statement must have a line number, the word 'INPUT' in upper case letters and the numerical variable(s).

(3) The user has to answer the question mark (?) as instructed. He/she has to hit the RETURN key after he/she has finished typing in the information, or the computer will shut off after a while.

Summary

(1) An INPUT statement is used to provide data value by the user of the program.
(2) The question mark (?) printed out by the computer asks the user to feed in information as instructed.
(3) In assigning values, only a numerical value can be assigned to a numerical variable and only a string value can be assigned to a string variable.
(4) Always use PRINT statement(s) before any INPUT statements to specify the instruction for the user.
(5) Use clear and unambiguous instructions in the PRINT statement to tell the user what he/she should be doing.

(6) The user should always do exactly as instructed.

Exercises

(1) What is the difference between LET and INPUT statements in assigning values to variables.

(2) Explain why PRINT statement(s) is (are) needed before INPUT statement.

(3) Write a program to ask user's name and age.

(4) Write a program to calculate the square of a number. Use INPUT statement to get user to feed in the number and the computer to type out the result.

Answers

(1) Both LET and INPUT statements function in the similar way. They are used to assign values to variables. But the value assignment in LET statement is done by the programmer who designs the program. In INPUT statement, the value is provided by the user when the program is executing. The value-assignment in INPUT statement changes because it depends on the input of the particular user.
(or equivalent answer)

(2) The INPUT statement is always preceded by PRINT statement(s) because the string constant in PRINT statement(s) serves as an explanatory message telling the user what to do. Clear and precise

instructions have to be used in PRINT statements to avoid ambiguities and misinterpretation. (or equivalent answer)

(3) Input program

```
10 PRINT "PLEASE TYPE YOUR NAME."
20 INPUT N$
30 PRINT "PLEASE TYPE YOUR AGE USING AN ARABIC NUMBER."
40 INPUT A
50 PRINT N$ ", IT IS NICE TO MEET YOU."
60 END
```

(or equivalent answer)

Output program

```
RUN
PLEASE TYPE YOUR NAME.
?JOHN BROWN
PLEASE TYPE YOUR AGE  USING AN ARABIC NUMBER."
?23
JOHN BROWN, IT IS NICE TO MEET YOU.
```

(4) Input program

```
10 REM THIS IS A PROGRAM TO CALCULATE THE SQUARE OF A NUMBER.
20 PRINT "PLEASE TYPE AN ARABIC NUMBER."
30 PRINT "HIT RETURN KEY AFTER YOU HAVE FINISHED."
40 INPUT N
50 PRINT "THE SQUARE OF YOUR NUMBER IS " ; N*N; "."
60 END
```

(or equivalent answer)

Output program

```
RUN
PLEASE TYPE A NUMBER IN ARABIC.
HIT RETURN KEY AFTER YOU HAVE FINISHED.
?6
THE SQUARE OF YOUR NUMBER IS 36.
```

Related concepts

LET statement (p.123), String constant (p.48), String variable (p.66), Numerical constant (p.38), Numerical variable (p.60)

SELF TEST

Note

If you can answer the following questions correct, you are through with this course unit. If not, go to page 160.

Exercises

(1) Write out the output of the following statements:-

```
10 READ A,B,C,D,E,F
20 DATA 1,2,3
30 DATA 4,5,6
40 PRINT A;B;C;D;E;F
```

(2) Study the following program:-

```
10 LET A=5
20 LET B=6
30 LET A$="APPLES"
40 LET B$="BIRDS"
50 PRINT A;A$, B;B$
```

Use READ & DATA statements to rewrite the above program.

Answers

(1)
```
RUN
123456
```

(2)
```
10 READ A,A$,B,B$
20 DATA 5, APPLES, 6, BIRDS
30 PRINT A;A$, B;B$
```

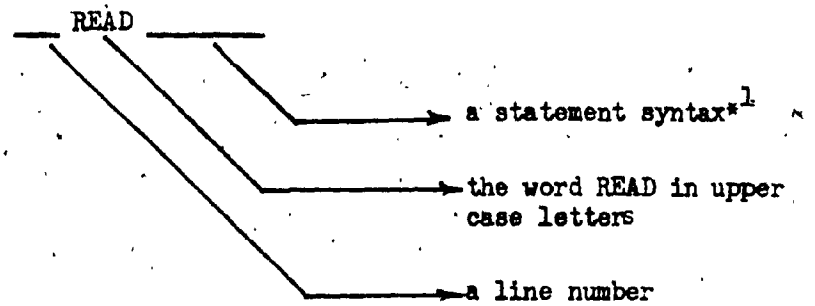
READ AND DATA STATEMENTS

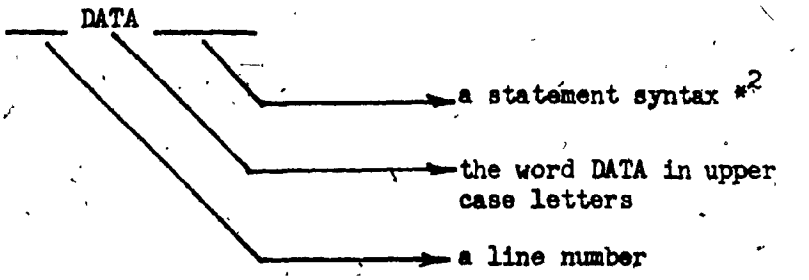
Introduction

The assignment of values to variables is a very important concept in computer programming. LET and INPUT statements are good examples. READ & DATA statements are also used to assign values to variables. They are extensions of LET statements.

What are they?

READ & DATA statements are twins because they always work together. They are used in the same way as LET statements. The LET statement is used to assign one value to one variable. But READ & DATA statements are usually used to assign a bunch of values to a bunch of variables. Their format is:





*Note: The two horizontal lines in the format of both READ & DATA statements are for illustrative purposes and they do not appear in your program.

*[1]Note: The statement syntax of the READ statement can be:-

(1) a variable, or

(2) more than one variable (use a comma to separate each variable).

*[2]Note: The statement syntax of the DATA statement can be:-

(1) a value (constant), or

(2) more than one value (constant) (use a comma to separate each value)

In a READ statement, you can find one or more variables, and, in a DATA statement, you can find one or more values (constants). The DATA statement is used to provide a number of values (or one value) for the variables (or one variable) listed in their respective READ statement(s). In other words, the READ statement respectively (i.e. in the same sequence as listed in both READ & DATA statements) assigns values (or one value) to the variables (or one variable) listed in the DATA statement.

Example 1 (A) Assign a numerical value(s)(constant) to a numerical variable(s)

Input program

```
10 READ B
20 PRINT B
30 DATA 505
```

Explanation of the input program

10 READ B

- a numerical variable
- the word READ in upper case letter
- This statement means: Let 'B' be assigned with the value of '505' listed in line 30. When the computer executes line 10, it will search for a DATA statement and store the value of '505' for the numerical variable 'B'.

20 PRINT B — This asks the computer to print out the value of 'B'.

30 DATA 505

- a numerical value (constant)
- the word DATA in upper case letter
- This DATA statement provides the value to the variable listed in READ statement.

Output program

RUN

505 → The value of 'B' is printed.

Note:

The above READ & DATA statements are equivalent to:-

Non-example 1 Input program

```
10  READ B
20  PRINT B
30  DATA FIVE HUNDRED AND FIVE
```

Output program

```
RUN
SN ERROR IN 30
```

Explanation

Notice B in line 10 is a numerical variable which only can be assigned with a numerical value. 'FIVE HUNDRED AND FIVE' in line 30 is a string value but '505' is a numerical value. That is the reason the computer prints SN ERROR IN 30.

Example 4 (B) Assign value(s) to string (textual) variable(s)

READ and DATA statements also can be used to assign string value(s) to string variable(s). There are 286 string variables we can use. It is a must to put a dollar sign (i.e. $) for every string variable we use (e.g. A$, Q$, A6$, M7$ etc.).

Input program

```
10  READ M$
20  PRINT M$
30  DATA FIVE HUNDRED AND FIVE
```

Output program

```
RUN
FIVE HUNDRED AND FIVE
```

Example 5

Input program

```
10  DATA FIVE HUNDRED AND FIVE
20  READ M$
30  PRINT M$
```

Output program

```
RUN
FIVE HUNDRED AND FIVE
```

Example 6

Input program

```
10  READ M$
20  DATA 505
30  PRINT M$
```

Output program

```
RUN
505
```

The above example shows numeral in Arabic form can be used as string value but string value cannot be assigned to numerical variable.

Example 7

(C) Assign a large amount of values

All the examples above show a READ statement taking one value from a DATA statement. Actually the main function of READ and DATA statements is to assign a number of values to a number of variables. This is the main advantage of READ and DATA statements. However, the LET statement can only assign one value to a variable at a time.

Input program

```
10 READ A,B,C,D
20 DATA 5,6,20,100
30 PRINT A;B;C;D
```

Output program

```
RUN
5,6,20,100
```

Explanation

10 READ A,B,C,D ——→ The numerical variables used are: 'A','B','C' & 'D'. Use a comma to separate all the variables.

20 DATA 5,6,20,100 ——→ The numerical values '5', '6', '20' & '100' are assigned to 'A', 'B', 'C' and 'D' respectively.

30 PRINT A;B;C;D ——→ Use a semicolon to separate each numerical variable in PRINT statement. (Still remember the use of a semicolon?)

As a matter of fact, the READ & DATA statements can be replaced by LET statements as follows:-

```
10 LET A=5
20 LET B=6
30 LET C=20
40 LET D=100
50 PRINT A;B;C;D
```

Obviously, by using READ & DATA statements you can avoid a lot of LET statements and at the same time it also provides you with the same result.

Example 8

Input program

```
10  READ A
20  READ B,C
30  READ D
40  DATA 5,6,20,100
50  PRINT A;B;C;D
```

Output program

```
RUN
5,6,20,100
```

Example 9

Input program

```
10  READ A,B,C,D
20  DATA 5,6,
30  DATA 20
40  DATA 100
50  PRINT A;B;C;D
```

Output program

```
RUN
5,6,20,100
```

Note:

The above three examples show the same result even when the programs are different. You can see that you can use several DATA and READ statements to list your values and variables. These multiple DATA statements and READ statements can also be condensed into one DATA statement and one READ statement when you input your program, the computer stores all the values of all DATA statements according to the line number sequence. When

it executes the READ statements, it will assign the values in the DATA statements one by one to the variables in the READ statement as follows:-

	Variable	Value
first item	A	5
second item	B	6
third item	C	20
fourth item	D	100

The first item of the READ statement 'A' is assigned the value 5, 'B' 6, 'C' 20 and 'D' 100. This is the way the computer reads your DATA and READ statements. Thus, special care must be taken to arrange the DATA values to be read in the sequence you want them to match the variables in READ statements.

Example 10 (D) Assign numerical and string values at the same time

Input program

```
10 DATA "HE HAS ", 5, " CHILDREN."
20 READ H$, A, D$
30 PRINT H$; A; D$
```

Output program

```
RUN
HE HAS 5 CHILDREN.
```

Explanation

```
10 DATA "HE HAS ", 5, " CHILDREN."
```

→ Leave a space here and there.

20 READ H$,A,D$ → 'H$' is assigned with the value "HE HAS ". 'A' is assigned with the value '5' and 'D$' is assigned with the value " CHILDREN."

30 PRINT H$;A;D$ → The computer is asked to print the values 'H$', 'A' and 'D$'.

The arrangement of the variables and values are shown below:-

Item	Variable	Value	Type of Variable
first	H$	HE HAS	string
second	A	5	numerical
third	D$	CHILDREN	string

Notice the use of " "(quotation marks) in line 10. The item enclosed in the quotation marks is considered to be one item. Since it is a string value, it must be assigned to a string variable while the numerical value is assigned to a numerical variable. Also notice in examples 4 and 5 that there are no quotation marks in the string variable. In this case, it is treated as one item.

Non-example 2 Input program

```
10  DATA "HE HAS ", 5, " CHILDREN."
20  READ A, H$, D
30  PRINT A;H$;D
```

Output program

```
RUN
SN ERROR IN LINE 20
```

How come there is syntax error in line 20? O.K. Let us see how the computer interprets this program:-

Item	Variable	Value
first	A	HE HAS
second	H$	5
third	D	CHILDREN

← wrong assignment

A in line 20 is a numerical variable and cannot take the string value "HE HAS ". D is also a numerical variable and cannot take the string value "CHILDREN." This program has to be debugged to assign numerical values to numerical variables and string values to string variables.

-Application Study the following program:-

Input program

```
10  READ A
20  DATA 5,6,10,20,50
30  PRINT A,A*A
40  GO TO 10
50  END
```

Output program

```
RUN
5             25
6             36
10            100
20            400
50            2500
OUT OF DATA IN LINE 10
```

Explanation

This program is designed to calculate the square of some numbers. There is only one numerical variable in line 10 i.e. 'A' but there are 5 different values assigned to 'A'. The numerical variable 'A' takes the 5 values (i.e. '5','6','10', '20' and '50') one at a time. That is to say each time the computer encounters statement 10, the numerical variable 'A' takes on a new value. When it uses up one value, it goes on to the next value and so on. 'A' takes its values in sequence as follows:-

Round	Numerical variable	Value
first	A	5
second	A	6
third	A	10
fourth	A	20
fifth	A	50

'A' takes its value one by one and this action is caused by the statement line number 40:-

40 GO TO 10 ⟶ This statement means jumping back to the line 10 and asks the computer to execute the operations before line 40. The operation will go on until it reaches the last value i.e. '50'.

Summary

1. READ & DATA statements are twins and they have to work together.
2. They are used in the same way as LET statements. LET is used to assign one value to one variable but READ & DATA can be used to assign a large number of values to a large (continued next page)

number of variables. They can be used to assign values to both numerical and string variables.

3. The READ statement is used to list a number of variables (or one variable) while DATA is used to list a number of values (or one value).
4. Only numerical values can be assigned to numerical variables and string values to string variables.
5. Always arrange all the values in the DATA statements in the way you want them to match the variables listed in the READ statements in your program.

Exercises

(1) Write out the output of the following statements:-

```
10 READ A,B,C,D,E,F
20 DATA 20,50,60
30 DATA 70,100,150
40 PRINT A;B;C;D;E;F
```

(2) Write a program to PRINT the following information by using READ & DATA statements:-

John's age: 20
Henry's age: 30
Jarry's age: 25

Answers

(1)

Output-program

```
RUN
20506070100150
```

(2) Input program

```
10  READ A$,A,B$,B,C$,C
20  PRINT A$;A
30  PRINT B$;B
40  PRINT C$;C
50  DATA "JOHN'S AGE: ", 20, "HENRY'S AGE: ", 30,
60  DATA "JARRY'S AGE: ", 25
```

Output program

```
RUN
JOHN'S AGE: 20
HENRY'S AGE: 30
JARRY'S AGE: 25
```

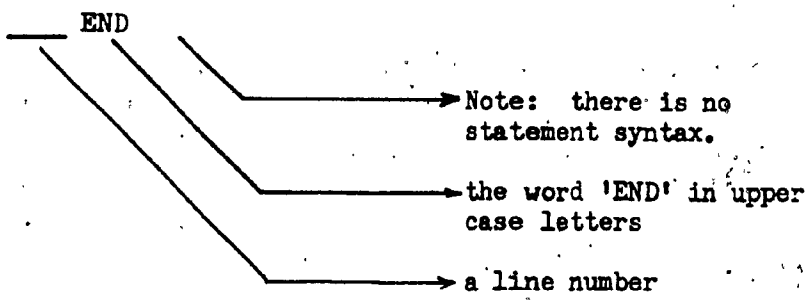
Related concepts

LET statement (p.123), **INPUT statement** (p.135), **Variable** (p.56), **Numerical variable** (p.60), **String variable** (p.66)

END STATEMENT

What is it?

The END statement is a very simple but essential statement. It has to be placed at the end of a BASIC program and it must have the highest line number in the program. Every program must have one and only one END statement. (*Note: In some BASIC systems, the END statement can be omitted.) Its function is to indicate the end of a BASIC program. When this statement is executed the computer simply stops executing the program. Its format is:-



*Note: The horizontal line in the format above is for illustrative purposes and it does not appear in your program.

Example 1

```
10 REM.....
20 REM.....
20 PRINT.....
30 LET.....
.
.
.
.
.
5000 END
```

REFERENCE: SYSTEM COMMANDS

*Note: Only two system commands RUN & LIST are enclosed here for reference purposes.

SYSTEM COMMAND: RUN

What is it?

RUN is a system command that asks the computer to execute the program for you. The computer will execute your program statement by statement in line sequence. You have to type the word 'RUN' in upper case letters, then hit the RETURN key, and the computer will start the operation.

Example 1

Input program

```
10 PRINT "HELLO!"
20 PRINT "HOW ARE YOU?"
```

Output program

```
RUN  ------------------------>  Type 'RUN' and hit
                                the RETURN key.
HELLO!
HOW ARE YOU?
```

Explanation

When the computer executes line 10, it prints out 'HELLO!'. When it executes line 20, it prints out 'HOW ARE YOU?'

Note

RUN system command directs computer to do all the operations in your program till the end of it.

Example 2

Input program

```
10 PRINT 3*6
20 PRINT 5+10
```

Output program

```
RUN
18
15
```

Explanation

```
10 PRINT 3*6 ----------> Here you ask the computer to give you the answer of 3 x 6.

20 PRINT 5+10 ---------> Here you ask the computer to calculate 5 plus 10.

RUN ------------------> You type the word 'RUN' and hit the RETURN key.

18
15 ------------------> The computer prints out the answers of the arithmetic expressions.
```

Note

The RUN system command also checks for syntax errors in the program. If there are syntax errors in your program the computer will print out the error messages at the end of your program after you have typed the word 'RUN' and hit the RETURN key. The computer refuses to execute your program until all the syntax errors have been debugged.

Example 3

Input program

```
10 PRINL "HELLO!"
20 PRINT "HOW ARE YOU?
```

Output program

```
RUN

ERROR LINE 10
ERROR LINE 20
```

Explanation

```
10 PRINL "HELLO!" ──────────► The word 'PRINT' is
                              misspelled.

20 PRINT "HOW ARE YOU? ─────► The closing quotation
                              mark is missing.

RUN ────────────────────────► You type the word 'RUN'
                              and hit the RETURN key.

ERROR LINE 10 }
ERROR LINE 20 } ────────────► The computer gives you
                              the error messages of
                              the wrong statements.
                              These statements have
                              to be debugged (retyped)
                              so that the computer
                              can execute the program
                              for you.
```

Related concepts: System command , LIST (p.179), RETURN
syntax error debugging

SYSTEM COMMAND: LIST

What is it?

LIST is a system command which you use to ask the computer to produce an exact copy of the input program that you have typed in. Of course you have to SAVE (another system command) your program before, otherwise there is no program in the memory cell. When you want to retrieve it you simply type the word 'LIST' and hit the RETURN key. You can ask the computer to LIST the whole program, some statements or just one statement in your program.

Example 1

Input program

```
10 PRINT "HELLO!"
20 PRINT
30 PRINT "HOW ARE YOU?"
```

Listed program

```
LIST
10 PRINT "HELLO!"
20 PRINT
30 PRINT "HOW ARE YOU?"
```

Explanation

You can see the input and the listed programs are exactly the same.

Note

Please note the difference between the RUN and LIST system commands. If you have doubt refer the preceding pages on RUN system command.

Example 2

Even if you do not type in your program according to the line number sequence, the computer will rearrange the line numbers in logical sequence and produce the whole program after you type the word 'LIST' and hit the RETURN key.

Input program

```
30 PRINT "HOW ARE YOU?"
10 PRINT "HELLO!"
20 PRINT
```

Listed program

```
LIST
10 PRINT "HELLO!"
20 PRINT
30 PRINT "HOW ARE YOU?"
```

Explanation.

```
30 PRINT "HOW ARE YOU?"  }
10 PRINT "HELLO!"        }   This is the program that you typed in, statement 30 first, statement 10 second, and statement 20 last.
20 PRINT                 }

LIST ------------------------> You type the word 'LIST' and hit the RETURN key.

10 PRINT "HELLO!"        }
20 PRINT                 }   Now look at all these statements. They are rearranged according to their line number sequence.
30 PRINT "HOW ARE YOU?"  }
```

Example 3

If you have a very lengthy program and you just want to LIST some statements in your program you can do as follows:-

Input program

```
100 PRINT "HELLO!"
110 PRINT
 .
 .
 .
500 LET T=3
510 LET Z=10
520 LET X=50
 .
 .
 .
1000 DATA 1,2,3,4,5,6
1010 END
```

Listed program

```
LIST 500,520
500 LET T=3
510 LET Z=10
520 LET X=50
```

Explanation

`LIST 500,520` → You type the word 'LIST', the line number that you want the computer to start to LIST, a comma, and the last line number that you want the computer to stop to LIST.

```
500 LET T=3
510 LET Z=10
520 LET X=50
```

After you have typed 'LIST 500,520' and hit the RETURN key the computer obeys your command and just lists those statements you want.

Example 4

If you want the computer to LIST from line 520 to the end of your program you can type:-

```
LIST 520, 1010
```

- LIST — the word 'LIST' in upper case letters
- 520 — the line number of the statement which you want the computer to start to LIST
- , — a comma
- 1010 — the line number of the last statement in your program

*Note: hit the RETURN key after you finished this line.

```
520 LET X=50
      .
      .
      .
1000 DATA 1,2,3,4,5
1010 END
```

The computer lists out the statements that you requested.

*Note: Some BASIC systems may have different features in requesting the computer to start listing at one line number and continue through at another line number.

Related concepts

System command , RUN (p.176), SAVE
line number (p.26)

REFERENCES ON BASIC

Alcock, D. Illustrating BASIC: A Simple Programming Language. Cambridge University Press, Cambridge, 1977.

Barnett, E.H. Programming Time-shared Computers in BASIC. Wiley-Interscience, New York, 1972.

Bennett, W.R. Introduction to Computer Applications for Non-science Students (BASIC). Englewood Cliffs, New Jersey: Prentice-Hall, 1976.

Bent, R.J. BASIC: An Introduction to Computer Programming. Books/Cole Pub. Co., Monterey, Calif., 1978.

Bitter, G.G. BASIC for Beginners. Wilson McGraw-Hill, 1978.

Brady, A.H. & Richardson, J.T. BASIC Programming Language. Learning Systems Co., 1974.

Brown, J.R. Instant BASIC. Dymax, Menlo Park, Calif., 1978.

Coan, J.S. Basic BASIC: An Introduction to Computer Programming in BASIC Language; Hayden Book Co., N.Y., 1970.

De Rossi, C.J. Learning BASIC Fast. Reston Pub. Co. Inc., Virginia, 1974.

Dwyer, T.A. BASIC & the Personal Computer. Addison-Wesley Pub. Co. Reading, Mass., 1978.

Dwyer, T.A. A Guilded Tour of Computer Programming in BASIC. Houghton Mifflin, Boston, 1973.

Farina, M.V. Programming in BASIC: The Time Sharing Language. Prentice-Hall, Englewood Cliffs, N.J., 1968.

Gear, C.W. BASIC Language Manual. Science Research Associate, Chicago, 1978.

Gosling, P.E. Beginning BASIC. Macmillan, London, 1977.

Hull, T.E. An Introduction to Programming & Applicationw with BASIC. Addison-Wesley Pub., Don Mills, Ont., 1979.

Kemeny J.G. & Kurtz, T.E. BASIC Programming. John Wiley & Sons, N.Y., 1967.

Ledin, G. A Structured Approach to General BASIC. Boyd & Fraser Pub. Co., San Francisco, 1978.

Lewis, R. & Blakeley, B.H. Elements of BASIC. National Computing Centre, Manchester, 1972.

Logsdon, T. Programming in BASIC. Anaheim Pub., Co., Ca., 1978.

Monro, D.M. Basic BASIC: An Introduction to Programming. E. Arnold London, 1978.

Morton, J.B. Introduction to BASIC. Matrix Pub., Champaign, Ill., 1977.

Moursund, D. BASIC Programming for Computer Literacy. McGraw-Hill Book Co., N.Y., 1978.

Mullish, H. A Basic Approch to BASIC. Wiley, N.Y., 1976.

Murrill, P.W. & Smith, C.W. BASIC Programming. Intext Educational Pub., Scranton, Pa., 1971.

Nevison, J.M. The Littler Book of BASIC Style: How to Write a Program You Can Read. Addison-Wesley Pub. Co., Reading, Mass., 1978.

Peckham, H.D. BASIC: A Hands-on Method. Mc-Graw-Hill, N.Y., 1978.

Peluso, et al. Basic BASIC: Self-Instructional Manual. Addison-Wesley, Reading, Mass., 1972.

Raskin, J. Apple II: BASIC Programming Manual. Apple Computer, Inc., 1978

Sack, J. & Meadows, J. Entering BASIC. Science Research Ass., Inc., Chicago, 1973.

Sharpe, W.F. BASIC: An Introduction to Computer Programming Using the BASIC Language. Free Press, N.Y., 1979.

Spencer, D.D. Using BASIC in the Classroom. Camelot Pub. Co., Florida, 1978.

Waite, S.V.F. & Mather, D.G. BASIC. University Press of N.E., 1971.

Walker, T.M. Fundamentals of BASIC Programming. Allyn & Bacon, Boston, 1975.

Worth, T. BASIC for Everyone. Englewood Cliffs, Prentice-Hall, N.J., 1976.

Wu, N.L. BASIC: The Time-sharing Language. W.C. Brown, Dubuque, Iowa, 1975.

APPENDIX 1: PRETEST

PRETEST

Read the following questions carefully and answer them as instructed.

Part 1: Write down T (for true) or F (for false) in the blanks for the followings:-

___ (1) The following is a BASIC statement:-
120 PRINT "GOOD MORNING."

___ (2) The following is not a BASIC statement:-
SAVE

___ (3) Every BASIC statement must have a line number.

___ (4) The use of DATA statement is to assign appropriate values to the variables listed in the READ statements.

___ (5) When we type in our statements on the terminal we have to type them according to the line number sequence.

___ (6) END statement can be placed in the middle of a problem.

___ (7) Sometimes a PRINT statement can have no statement syntax.

___ (8) INPUT statement is used to assign value to a variable during the time the user is running the program. Its value is not provided by the program designer.

___ (9) In the following statement, '70' is a numerical constant:-
500 LET B$="70"

(10) 500 LET B=70
In the above statement, '70' is a numerical constant.

(11) 210 PRINT "700"
In the above statement, '700' is a string constant.

(12) 500 LET B$="70"
In the above statement, 'B$' is a numerical constant.

(13) 10 LET N$="HENRY"
20 PRINT N$
In the above two statements, 'N$' is a string variable.

(14) We can assign numerical values to numerical variables.

(15) 70 LET A=5
In the above statement, 'A' is a numerical variable.

(16) The following statement is correct:-
510 LET H*=750

(17) The following statement is correct:-
510 LET H2=750

(18) The following statement is incorrect:-
980 LET $A="NAME"

(19) The following statement is incorrect:-
980 LET A="NAME"

(20) The following statement is correct:-
980 LET A$="NAME"

Part 2: Match the items in (i) with those in (ii) in the blanks that you think most appropriate:-

	(i)	(ii)
___	(1) ↑	(a) means division
___	(2) 5*5	(b) means exponentiation
___	(3) 5E-7	(c) means multiplication
___	(4) *	(d) a floating point notation
___	(5) /	(e) an arithmetic expression
___	(6) C<>D	(f) C is equal or less than D.
___	(7) C=<D	(g) a system command
___	(8) RUN	(h) means squeezing print space together in PRINT statement
___	(9) ,	(i) means there are 5 tab (print) zone to move into
___	(10) ;	(j) C is not equal to D.

Part 3: Study the following two programs and write down what the output result (the printout result) would look like:-

Program 1

```
200 LET A=50
210 LET B=100
220 PRINT A*B
230 PRINT B/A
240 PRINT A,B
250 PRINT A;B
260 PRINT "A","B"
270 PRINT "AB"
280 PRINT B-A
290 PRINT A+B
300 END
```

Part 3

Program 2

```
510 PRINT "PLEASE TYPE VALUES FOR A,B,C & D."
520 PRINT "USE COMMA TO SEPARATE EACH VALUE."
530 INPUT A,B,C,D
540 LET X=(A+B+C+D)/4
550 PRINT "THE AVERAGE FO THE 4 NUMBERS IS"; X; " ."
560 END
```

Part 4: Find out the bugs (errors) in the following program. Then debug them and write them out correctly.

```
150 PRANT "THIS IS A PROGRAM TO FIND OUT"
160 PRINT "THE CUBE OF SOME INTEGERS.
170 PRINT "PLEASE TYPE A NUMBER IN ARABIC.
180 INPUT N$
PRINT "THE CUBE OF N$ IS", N$↑3; " ."
END
```

Part 5: Design BASIC programs for the followings:-

(1) Print 'A' at the first print position, 'B' at the 10th print position, and 'C' at the 20th print position.

(2) Write a short BASIC program that will print out your name and address.

Part 5:

(3) Write a BASIC program asking the user to print out his/her name and address on the terminal. Store them in the computer memory.

(4) Write a program to print the values of 'A','B','C' & 'D' on the same line with 5 printing space apart where

A=20
B=40
C=50
D=60

Use READ & DATA statements in your program.

(5) Study the following program:-

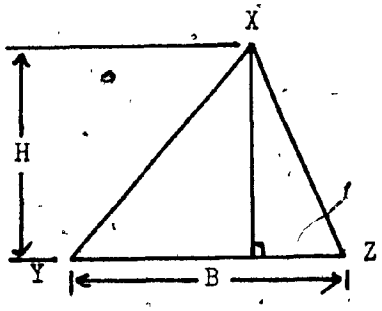
```
40 PRINT "PLEASE TYPE 3 VALUES FOR X,Y, & Z."
50 PRINT "USE COMMA TO SEPARATE EACH VALUE."
60 INPUT X,Y,Z
70 LET S=X+Y+Z
80 PRINT "THE SUM OF THE 3 NUMBERS IS"; S; "."
90 END
```

Rewrite the above program by using READ & DATA statements instead of INPUT statement (line 60). Make necessary changes if needed.

Part 5:

(6) The area of a triangle is calculated:-

Area of $\triangle XYZ = \frac{\text{Height x Base}}{2}$



H = The height of a triangle.

B = The base of a triangle

Write a program to calculate the area of a triangle. The program should ask input values for the height and the base of the triangle. (That is to say the program user is going to supply these data.)

Part 5:

(7) Design a simple program for a cashier to calculate the change for a customer when he/she pays his/her bill in your supermarket. The 3 factors involved in this program are:-

(a) The total amount of purchases,

(b) The amount paid by the buyer, and

(c) The change given to the buyer.

Part 5:

(8) Write a program that will read the values of A, B & C and print them out on one line with no printing space separating them. Their values are:-

A=20
B=30
C=40

Also ask the computer to print out:-

(a) the sum of all the values,
(b) the multiplication of all the values,
(c) the answer of C minus A, and
(d) the answer of C divided by A.

APPENDIX 2: POSTTEST

Date: 26th June, 1980.

Score:

POSTTEST

Read the following questions carefully and answer them as instructed.

Part 1: Write down T (for true) or F (for false) in the blanks for the followings (Note: Wrong answers will be penalized.):-

___ (1) The following is a System Command:-
RUN

___ (2) The following is a BASIC statement:-
500 LET A=700

___ (3) Some BASIC statements have no line number.

___ (4) READ statement is used to provide a number of values (or one value) for the variables (or one variable) listed in DATA statement respectively.

___ (5) Two BASIC statements can have the same line number.

___ (6) END statement is placed at the end of a program.

___ (7) It is a syntax error if there is no statement syntax for PRINT statement.

___ (8) The value-assignment of INPUT statement is given at the time when the program is running.

(9) 800 LET Z$="50"
In the above statement, '50' is a numerical constant.

(10) 800 LET Z=50
In the above statement, '50' is a numerical constant.

(11) 70 PRINT "100"
In the above statement, '100' is a string constant.

(12) 90 LET B$="100"
In the above statement, 'B$' is a numerical constant.

(13) 50 LET Y$="YES"
60 PRINT Y$
In the above 2 statements, 'Y$' is a string variable.

(14) We can assign string values to numerical variables.

(15) 150 LET Z5=8
In the above statement, 'Z5' is a numerical variable.

(16) The following statement is correct:-
270 LET H#=500

(17) The following statement is correct:-
780 LET Z8=8

(18) The following statement is correct:-
890 LET $H="MALE"

(19) The following statement is correct:-
100 LET F="FAILURE"

(20) The following statement is correct:-
500 LET G$="FAILURE"

Part 2: Match the items in (i) with those in (ii) in the blanks that you think most appropriate.

	(i)	(ii)
___	(1) X<Y	(a) X is greater than Y.
___	(2) LIST	(b) means exponentiation
___	(3) *	(c) a floating point notation
___	(4) X>Y	(d) X is less than Y
___	(5) /	(e) means multiplication
___	(6) 5+70-20/2↑5	(f) means to separate 5 printing spaces
___	(7) ,	(g) means no printing space
___	(8) ↑	(h) a system command
___	(9) ;	(i) an arithmetic expression
___	(10) 8E-9	(j) means division

Part 3: Study the following 3 programs and write down what the output result (the printout result) would look like:-

Program 1

```
500 LET F=80
510 LET G=10
520 PRINT F
530 PRINT G
540 PRINT F+G
550 PRINT F*G
560 PRINT F/G
570 PRINT "F", "G"
580 PRINT "F"; "G"
590 PRINT "FG"
600 END
```

Program 2

```
700 READ X,Y,Z
710 LET A=X+Y
720 LET B=Z+A
730 PRINT A
740 PRINT B
750 PRINT X,Y,Z
760 DATA 4,2,5
770 END
```

Program 3: (Assume that you are the user of this program and write out its output result.)

```
500 REM This program is designed by J. Roberts.
510 PRINT "Please type values for W,X,Y & Z."
520 PRINT "Use comma to separate each value."
530 INPUT W,X,Y,Z
540 LET A=(W+X+Y+Z)/4
550 PRINT "The mean of the inputted values is ";A;" ."
560 END
```

Part 4: Find out the bugs (errors) in the following program. Then debug them and write them out correctly:-

```
10 PRINT "THE NAME OF THE PROGRAM IS XABC."
20 PIANT "IT IS DESIGNED BY JOE ROBERTS.
LET A=NAME
LET B=SEX
50 PRINT PLEASE TYPE YOUR NAME."
60 INPUT A
70 A ", IT'S NICE TO MEET YOU.'
```

.....

Part 5: Code the followings in BASIC:-

(1) Print 'Name' at the 5th print position, '*' at the 15th print position, and '#' at the 30th print position (on one line).

(2) Write a short program to print the following information:-
1st line: Hello!
2nd line: How are you?

(3) Write a BASIC program asking the user to print out his/her name and age on the terminal. Store them in the computer memory.

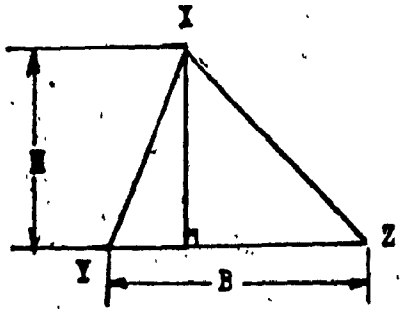
(4) Write a program to print out the values 'W', 'X', 'Y' & 'Z' on the same line with 5 printing spaces apart where

W=10
X=100
Y=500
Z=50

Use READ & DATA in your program. Don't use LET statement.

(5) The area of a triangle is calculated:-

$$\text{Area of } \triangle XYZ = \frac{\text{Height} \times \text{Base}}{2}$$



H=the height of the triangle

B=the base of the triangle

Write a program to calculate the area of a triangle. The program should ask input values for the height and the base of a triangle. (i.e. The user of the program is going to supply these data.)

(6) Design a simple program for the cashier to calculate the change for the customer when the customer pays his/her bill in your supermarket. The 3 factors involved in this program are:-

(a) the total amount of purchases,

(b) the amount paid by the buyer, and

(c) the change given to the buyer.

(7) Write a program that will read the values of A,B, & C and print them out on one line with no printing space separating them. Their values are:-

A=20
B=30
C=40

Also ask the computer to print out:-

(a) the sum of all the values,
(b) the multiplication of all the values,
(c) the answer of C plus A, and
(d) the answer of C divided by A.

MARKING SCHEME FOR POSTTEST

Part 1: 10% (i.e. 0.5 for each question)

(1) T
(2) T
(3) F
(4) F
(5) F
(6) T
(7) F
(8) T
(9) F
(10) T
(11) T
(12) F
(13) T
(14) F
(15) T
(16) F
(17) T
(18) F
(19) F
(20) T

Part 2: 5% (i.e. 0.5 for each question)

d (1)
h (2)
e (3)
a (4)
j (5)
i (6)

f (7)
b (8)
g (9)
c (10)

Part 3:

Program 1: 9% (1 mark for each item)

80
10
90
800
8
F G
FG
FG

Program 2: 5%

6
11
4 2 5

Program 3: 8%

Please type values for W,X,Y & Z.
Use comma to separate each value.
?1,2,3,4
The mean of the inputted values is 2.5
(or equivalent answer)

Part 4: 11%

```
10 PRINT "THE NAME OF THE PROGRAM IS XABC."
20 PRINT "IT IS DESIGNED BY JOE ROBERTS."
30 LET A$="NAME"
40 LET B$="SEX"
50 PRINT "PLEASE TYPE YOUR NAME."
60 INPUT A$
70 PRINT A$; " IT'S NICE TO MEET YOU."
```

Part 5:

(1) 2%

```
10 PRINT TAB(5); "NAME" TAB(15); "*" TAB(30) "#"
```

(2) 2%

```
10 PRINT "HELLO!"
20 PRINT "HOW ARE YOU?"
```

(3) 4%

```
10 PRINT "PLEASE TYPE YOUR NAME."
20 INPUT N$
30 PRINT "PLEASE TYPE YOUR AGE IN ARABIC NUMBER."
40 INPUT A
```

(or equivalent answer)

(4) 8%

```
10 READ W,X,Y,Z
20 DATA 10,100,500,50
30 PRINT W,X,Y,Z
40 END
```

(5) 12%

```
10 REM IT IS A PROGRAM TO CALCULATE THE AREA
20 REM OF A TRIANGLE.
30 PRINT "PLEASE TYPE THE HEIGHT OF A TRIANGLE IN ARABIC."
40 INPUT H
50 PRINT "PLEASE TYPE THE BASE OF THE TRIANGLE IN ARABIC."
60 INPUT B
70 LET A=(H*B)/2
80 PRINT "HEIGHT", "BASE", "AREA"
90 PRINT H,B,A
100 END
```

(or equivalent answer)

(6) 12%

```
10 REM IT'S A PROGRAM FOR CALCULATING CHANGES.
20 PRINT "WHAT IS THE TOTAL AMOUNT OF PURCHASES?"
30 INPUT T
40 PRINT "WHAT IS THE AMOUNT PAID BY THE BUYER?"
50 INPUT B
60 LET E=B-T
70 PRINT "THE CHANGE IS";E; " ."
```

(or equivalent answer)

(7) 12%

```
10 LET A=20
20 LET B=30
30 LET C=40
40 PRINT A;B;C
50 PRINT A+B+C
60 PRINT A*B*C
70 PRINT C+A
80 PRINT C/A
90 END
```

APPENDIX 5: THE COURSE UNIT EVALUATION FORM

Name:

Date: 26th June, 1980.

COURSE UNIT EVALUATION

Read the following questions carefully and check the item that best suits your opinion. Please write down your comments if necessary. Thanks very much for your cooperation.

1. Did you finish the course unit?

   (a) Yes (b) No

2. How much time did you spend on the course unit?

   _____ hours

3. Did you use other books as reference?

   (a) Yes (b) No

4. If yes, how many books did you use? Please write down the name(s) of the book(s)?

   _____

   _____

5. If yes, how much time did you spend on the book(s)?

   _____ hours

6. Are the behavioral objectives clearly stated in the course unit?

very clear	5	4	3	2	1	not clear

Comments: _____

_____

_____

7. Is the content of the course closely related to the behavioral objectives?

related closely 5 4 3 2 1 many irrelevancies

Comments: _____

_____

_____

8. Is the content organized to promote learning?

well organized 5 4 3 2 1 poorly organized

Comments: _____

_____

_____

9. Does the text provide a clear picture of the subject treated?

very clear 5 4 3 2 1 not clear

Comments: _____

_____

_____

10. Is the language used clear and understandable?

very clear & understandable 5 4 3 2 1 not clear & understandable

Comments: _____

_____

_____

11. Are the examples and non-examples in the course unit helpful in explaining the concepts concerned?

very helpful 5 4 3 2 1 not helpful

Comments: _____

_____

_____

12. Does the format of the course unit (e.g. the label and the horizontal lines) encourage easy scanning?

very easy 5 4 3 2 1 not easy

Comments: _____

_____

_____

13. Are the graphic cues (e.g. arrows & underlinings) helpful in enhancing learning through emphasis and attention focusing?

very helpful 5 4 3 2 1 not helpful

Comments: _____

_____

_____

14. Do you prefer the format of this course unit to the traditional textbook?

very much 5 4 3 2 1 not at all

Comments: _____

_____

_____

15. Is the course unit helpful as self-instructional material?

very useful 5 4 3 2 1 not useful

Comments: _____

_____

_____

16. Name the best part and the strengths of this course unit:-

_____

_____

_____

_____

17. Name the worst part and the limitations of this course unit:-

18. Which part of the course unit would you like to keep?

19. Which part of the course unit would you like to get rid of?

20. What areas of the course unit would you like to explore further?